Building-block solutions for developing instructional software.

Building-block solutions for developing instructional software

Eddy Boot

Open Universiteit Nederland
Heerlen, 2005

OpenUniversiteitNederland

Het in dit proefschrift beschreven onderzoek werd uitgevoerd bij TNO Business Unit Human Factors in Soesterberg

The research reported in this dissertation was carried out at TNO Business Unit Human Factors in Soesterberg

Druk	Print Partners Ipskamp Enschede
Omslag	Nieke Janssen
Typografie advies	Leander Teepe

ISBN-10: 9090200584
ISBN-13: 9789090200583

e-mail: eddy.boot@tno.nl

Building-block solutions for developing instructional software

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Open Universiteit Nederland
op gezag van de rector magnificus
prof. dr. ir. F. Mulder
ten overstaan van een door het
College voor promoties ingestelde commissie
in het openbaar te verdedigen

op vrijdag 9 december 2005 te Heerlen
om 15.30 uur precies

door
Eduardus Willem Boot
geboren op 25 mei 1969 te Ommen

Promotor:
Prof. dr. J.J.G. van Merriënboer, Open Universiteit Nederland

Overige leden beoordelingscommissie:
Prof. dr. A.S. Gibbons, Brigham Young University
Prof. dr. J. Lowyck, Katholieke Universiteit Leuven
Dr. Y.F. Barnard, Eurisco
Prof. dr. P.A. Kirschner, Open Universiteit Nederland
Prof. dr. E.J.R. Koper, Open Universiteit Nederland

Acknowledgements

This dissertation is about improving the development of instructional software. Three solutions are proposed to assist instructional designers in communicating their designs to software producers better. The first solution is to support the designers in making a better use of formal *design languages* to standardize their design documents. The second solution is to support the designers by using *templates* that offer pre-structured software moulds that only have to be filled in with learning materials to create instructional software, either as examples for producers, or even as final products. And the third solution is to support designers in *reusing* parts of instructional software to assemble new instructional software, as examples for producers or as final products.

These solutions can be demonstrated in this Acknowledgement section. A study of several dissertations shows that most Acknowledgement sections are structured in a very uniform way, all using a similar design language. This implies that a structured template can be made, in which each user only has to instantiate his or her text. The result is a faster creation of this section, with the potential of reusing parts for other Acknowledgement sections! In this example, the eXtended Markup Language (XML) is used as a standardized design language, with a template-like structure, identifying multiple parts that can be reused, perhaps with some reediting. This example may also demonstrate potential drawbacks of the three building block solutions: Will they limit creativity?

<lom xmlns="http://www.imsglobal.org/xsd/imsmd_v1p2">
<general>
<title>
<langstring xml:lang="x-none">Acknowledgements</langstring>
</title>
<purpose>
<langstring xml:lang="x-none">Thanking everybody who has contributed to this dissertation</langstring>
</purpose>
</general>
<promoter>
Jeroen, for all his support, above and beyond duty, in conducting the studies and writing this dissertation. The endless discussion, writing, and lunch-walking sessions in Heerlen will be remembered with delight. It was Stoof (2005) who already identified one of the characteristic Four Components of Jeroen, namely “cognitive clearness”. I would like to add to that the Component of “cognitive meticulousness”, and leave it to my successors to identify the two remaining Components (or…perhaps there are more than four?).
</promoter>

<management>
Herke, Alma, Hans, Bernd, and Rick, successively my department heads, for the freedom they provided me to write this dissertation and allowing me to practice what we preach as a department, namely that learning must be integrated with working.
</management>
<co-authors and reviewers>
Jeroen, Jon, Andy, Arja, and Nicolet for their cooperation in recording my wild and divergent ideas on paper in such a way that also normal people will understand them. The reviewers Craig, Allard, Nicolet, Daniëlle, Marcel, Martijn, and Luca for wading through large amounts of text and providing valuable input. Jennifer and Roxana for the English corrections. Ergo: If this dissertation is still incomprehensible, it must be me.
</co-authors and reviewers>
<support>
Arno, Martin, Martijn, Jim, and Nicolet for their statistical advice, they all were significant contributors. John, Yvonne, and Arja for their successive coaching roles when this dissertation project started. I hope it is just coincidental that each of them left TNO shortly after they started coaching me... I want to thank all my colleagues at the department of Training & Instruction, supported by the tireless Dianne. Particularly, I want to thank the members of our former Learning Processes group: Gerard, Nicolet, Daniëlle, Arja, Martijn, Cathy, Sietske, Nieke, John, and Yvonne. Koos and Walter for the many multimedia issues they have solved for me over the years. Bart for the great pictures of the 3D-model. Allard for his technical support during the studies and Allard and Martijn for their lay-out advice. And finally, Nieke for her excellent cover design.
</support>
<studies>
Participants in the studies were military and civilian personnel from the Royal Netherlands Army (RNLA) and the Royal Netherlands Air Force (RNLAF), and students from the Computer Science Department from the Utah State University. I want to thank them all for bearing with me, while I gave them impossible tasks. The designers/producers from 367 TRSS/TSIT at Hill Air Force Base in Ogden, Utah helped piloting instructional materials, made possible by Deputy Flight Chief Embry and TSgt McIntyre. Furthermore, many people helped me in organizing the studies. First, internship students Christa and Maarit showed great organization and research skills in conducting their studies. From the RNLA, John, Markwin, Brenda, Robert, Ton, and Rien were of great value. From the RNLAF, I have received much support from Willem, Ton, Jan, Mattie, and Rob. Last but not least, Michael provided a lot of experimental design advice and practical input.
</studies>

<the-year-abroad>

TNO, particularly Alma, Jan H, Jan V, and Ronald, who made it possible to finish this dissertation at the Utah State University (USU) in Logan, Utah. Byron and the staff of USU's Instructional Technology department for organizing everything very well. Jon for making it a year to remember forever, academically, politically, and personally. Andy for his supervision from Brigham Young University in Provo, Utah. For our 'best one year' in Utah, Rian and I want to thank everybody who has made it possible for us to still consider Logan as our "home town": the Black's, the Kennedy's, the Merrill's (the whole clan that is), the Sweat's, the Nelson's, the King's, Van Schaack's, the Olson's, and the Eastman's.

</the-year-abroad>
<family>

When I left elementary school, the school master advised me to follow my father's footsteps, and become a farmer as soon as possible. The formal school tests (i.e., the CITO tests) found me to be more stubborn than intelligent, and joined this advice. They're foresight was excellent: Today's complex agricultural situation in the Netherlands requires at least a Ph.D. title to manage! Nevertheless, it was my parents who ignored the advice (you see where my stubbornness comes from!) and pushed me through a seemingly never-ending series of studies. I want to thank them for this enormous effort, and I hope I make them very proud with this book. I want to thank the rest of our families and friends for their love and support. Finally, I want to thank Rian, just for being with me…now it is time to search for new horizons to explore!

</family>
<conclusion>

Writing a dissertation is a long and complex endeavor. However, to quote a not unfamiliar song: "You'll never walk alone". In fact, I now see that writing this dissertation is similar to the topic studied in this dissertation: Developing instructional software. The task of the designer of both instructional software and a dissertation is to communicate his or her ideas clearly to an audience. I hope that this dissertation, as my latest development project, is successful in that respect.

</conclusion>

Index

Chapter 1

Introduction

Abstract

This introduction consists of five parts. First, the central problem of this dissertation, namely, the difficulties instructional designers experience in ensuring an unequivocal interpretation of their designs by software producers, is introduced by means of an illustrative scenario. Second, this "transition bottleneck" is further explored and, third, possible solutions are proposed, emphasizing new roles for instructional designers who have to interact more and more with the building blocks of the development process normally intended for software producers rather than instructional designers. Fourth, the main research questions of this dissertation are formulated, directed at testing three building-block solutions that support instructional designers in their new roles: (1) The Developing Design Documents (3D) model to support designers in stratifying, elaborating, and formalizing design documents; (2) instructional software templates to support designers in producing software themselves, and (3) an integrative approach to support designers in reusing learning objects. Finally, the structure of the dissertation is presented by briefly describing each of the four reported studies.

An Illustrative Scenario

"Willem, a staff employee at the training division of a large company, has just finished writing the new company policy document about developing instructional software. With his academic background in educational technology, he is always very interested in the latest innovations in his field. In his policy document, he has taken into account two recent innovations he thinks are particularly useful for his company. The first innovation is the increased focus on authentic learning tasks, based on real-life tasks, as the driving force for learning. Such tasks help learners to integrate the knowledge, skills, and attitudes necessary for effective task performance; give them the opportunity to learn to coordinate constituent skills that make up complex task performance, and eventually enable them to transfer what is learned to their daily life or work settings. Especially transfer to new situations is necessary within Willem's company because it is absolutely impossible to train employees for each new system and each new task, introduced at an ever-increasing rate. The second innovation Willem has included in the policy document is the application of blended learning or integrated e-learning: The combination of face-to-face learning, distance learning, and on-the-job learning. Blended learning is supported by a balanced media-mix of traditional and advanced learning technologies such as books, e-learning, mobile learning, and simulations. Implementing both innovations should provide Willem's company with the flexibility to promote the integration of working and learning, in terms of time- and place independent learning, as well as adaptive learning, personalized for individual learners with different backgrounds.

It is one week later. Jon, an instructional designer, works in the training development department of the same company as Willem. He has just received two documents. The first document is an assignment from the company's Technical Maintenance division to create a new instructional software product for training maintenance engineers for a complex system the company is about to introduce. The second document describes the new company training policy prepared by Willem. Jon briefly reads the first document and already starts to sketch in his mind a possible design for the new product. After that, he reads Willem's policy document. As a developer, he is particularly interested in the consequences of the new innovations, introduced by Willem, for the development processes and products. First, for the development process it appears that the new focus on authentic learning tasks makes it more important than ever to describe the task environment and the learning environment sufficiently detailed in design documents, which help to properly transfer this information to software producers. Second, for the development products, it appears that future instructional software packages must become much more flexible than they used to be. This flexibility is expressed in terms of adaptivity towards individual learners, generativity to assemble new configurations easily, and scalability to quickly expand and enlarge the possible target groups. Jon realizes that this will lead to a higher complexity of the instructional design process, and that he faces the difficult task to make his design documents as complete, concrete, and clear as possible.

Together with his team of instructional designers, Jon starts to create the new instructional design for the maintenance engineer training. Based upon the Four Components Instructional Design model they normally use in their training

development department, the designers start with an analysis of the task domain and the target group, in order to establish a task hierarchy and a global training sequence. Also, they analyze the task domain in terms of systems and the task environment of the maintenance engineers. Subsequently, they create the instructional design based on the domain representations and selected training strategies, and record this in a training blueprint. In addition, they provide guidelines for the implementation of the training blueprint in the instructional software by means of a storyboard. The storyboard contains sketches of the screen layouts and guidelines for media selection and navigation. After both types of design documents are finished, representing the total instructional design, Jon sends them to his project leader. The project leader will make a Request for Proposal (RFP) based upon these documents. This RFP will contain a brief description of the design requirements as well as other demands, and invite external production companies to submit a proposal. As usual, this proposal is based upon a fixed price: The external company must promise to create the whole instructional software program for the price stated in their reply to the proposal.

Michael works for a multimedia company creating instructional software for other companies. Michael's company submitted the lowest offer in the bidding process, and now he and his team will produce the new instructional software product for training maintenance engineers. He has received Jon's two design documents, and Michael starts reading the training blueprint and the storyboard. Quickly, he more or less foresees what kind of product he has to make, and how he should do it. He will start with creating the technical specifications. However, some issues bother him. First, he notices there are several, well-explained instructional design concepts in the training blueprint, such as feedback, modeling examples, scaffolding, and just-in-time information presentation items. However, the relations between these instructional design concepts and the instructional software concepts are not particularly clear. For instance, if the design documents propose video clips to represent a particular learning task and Michael wants to use photographs, which are cheaper to make, he doesn't know if and how such a static representation instead of a dynamic representation will affect the instructional strategies and thereby the learning outcomes. Second, parts of the instructional design lack sufficient detail. For Michael, the application of delayed cognitive feedback following a particular learning task, for instance, requires highly detailed descriptions of timing, content, and presentation to be able to specify and implement the feedback as intended by the instructional designer. Third, Michael notices that the textual descriptions in the training blueprint and the sketches in the storyboard leave him considerable room for own interpretation. This provides him some freedom in production choices, but also the danger that his product will not reflect Jon's design intentions. Furthermore, there is the danger of inconsequent translations of design principles to specifications. For example, similar types of delayed feedback may be interpreted differently in various parts of the design, resulting in different implementations of the same feedback in the product. In summary, Michael would have preferred (a) a rigorous one-to-one relation between design and software issues, (b) with a higher level of detail in the design descriptions, and (c) using notation systems that are less ambiguous. Note that Michael feels responsible for a good implementation of the design in the resulting products. Other software producers may not even think about these issues, and produce what is feasible within their budget. Such producers will think

they understand the design completely, whilst they are often only loosely interpreting the design. Nevertheless, Michael resolves to make the best of the situation. He sends Jon's design documents to his production team, and opens the group agenda to plan the kick-off meeting to start the technical specification process."

The scenario above demonstrates an important problem in the process of modern instructional software development, namely, the distortion and/or loss of relevant design information between the design and production phases – a phenomenon that will be referred to as the "transition bottleneck". Due to the criteria set by educational innovations, such as using authentic tasks and increasing flexibility, this problem becomes more prominent than ever before. It appears to be difficult for instructional designers to describe their design in such a way that unequivocal interpretation by software producers is ensured. This introductory chapter first presents an exploration of this problem. Second, possible solutions are proposed. Third, the main research questions of this dissertation are formulated. Finally, a brief overview of the dissertation is given.

Problem Exploration

New criteria for instructional software development, such as the use of authentic learning tasks (see Merrill, 2002; Reigeluth, 1999) and high flexibility (see Jochems, van Merriënboer, & Koper, 2004), pose high demands to the design and development process. To improve the efficiency of developing instructional software, *lean production* has been introduced (Woll, 2003). Lean production originates from the manufacturing industry and is a development approach that is directed at producing a broad variation of products, which are flexibly to adapt to individual users. This "mass-customization" should ensure adaptivity, generativity, and scalability of the instructional software products, satisfying the main features of flexibility. However, lean production does not solve the transition bottleneck, that is, the distortion and loss of information between the design and production phase. In contrast, lean production may make this problem even more prominent. One of the basic principles of lean production is that design information should be optimally interrelated to production information by placing emphasis on the manufacturability of the product as early as possible. This implies that instructional designers rather than software producers are actually responsible for overcoming the design-production transition bottleneck, and should try to accommodate to the information needs of software producers.

A fundamental problem in the transition between the design and production phase is the lack of standardized *design languages* that are familiar to both instructional designers and software producers. In other domains than the instructional design field (e.g., architecture, music, mechanics), such powerful design languages are able to capture and describe the design (i.e., blueprints, storyboards) with such a level of detail that software producers will interpret it unequivocally (Waters & Gibbons, 2004). But as a result of the lack of appropriate instructional design languages, it is very difficult for an

instructional designer to transfer his design and make sure that different software producers will interpret it in the same way, reach the same technical realization, and make the same calculation for costs and time necessary for producing the instructional software. This is especially important in the case of outsourcing, but also relevant for project calculations and task allocation decisions for production groups within the same organization.

Of course, there are alternative ways to improve the transfer of information from the design to the production phase. For instance, a software producer might decide to discuss indistinctnesses in the blueprint and storyboard with the instructional designer(s) during the drafting of technical specifications. Or the software producer might decide to apply a rapid prototyping approach and frequently show prototypes to the instructional designer to trigger discussions. However, both approaches are often undesirable from a project management point of view. Discussion costs considerable time. Prototyping will cost even more time, and evaluation of instructional software prototypes with learners is often difficult. Mostly, such extra time expenditures are not included in the initially set price. Also, during discussing or reviewing prototypes, instructional designers will likely give suggestions to the software producers that change and embellish the design and thus expand the necessary production activities.

Ideally, the design should be transferred from the instructional designer or design team to the software producer or production team only once, and be completely understood by the software producer(s). This way, there is either no further information exchange necessary, or developers can formulate clear and concrete questions about, for instance, details of the task domain. Thus, an ideal design should allow the production company to make an exact estimation of costs and time (before the contract is signed), and ensure a product that is fully compatible with the original design (after the contract is signed). A good solution should focus on supporting instructional designers to provide software producers with exactly the product-related information they need. The focus on instructional designers rather than software producers or other stakeholders is important, because the designers are pre-eminently responsible for the didactical quality of the final products (defined as the extent to which desired learning outcomes are attained in an efficient manner). This didactical quality is of utmost importance because technical quality (defined as the extent to which the software takes care of the input, information processing, and output as intended, and the responsibility of the producers) alone is a necessary but not sufficient condition to stimulate the desired learning processes and reach intended learning outcomes.

Possible Solutions

Currently, production-related information is typically embedded in three types of building blocks for the production process: (a) design documents as input; (b) programming structures as throughput, and (c) learning materials as

output. In order to better interrelate the design phase with the production phase, it is proposed to focus instructional designers' attention on these three particular types of building blocks. First, instructional designers could be supported to create *design documents* that are better organized, more detailed, and standardized according to particular formalisms. This should ensure that software producers are confronted with one-to-one relations between instructional design aspects and software aspects, the right level of detail, and unambiguous notation systems. Second, instructional designers could be supported in creating the *programming structures* (i.e., the software code) that make up didactically sound instructional software themselves, without the involvement of software producers. Modern authoring tools enable the quick and easy development of programming structures for instructional software, and are often used by software producers to make the production process more efficient. However, instructional designers can also choose to create these programming structures themselves in stead of the producers, because modern authoring tools require no or little technical programming skills. Third, instructional designers could be supported in reusing *learning objects*, another activity that is normally performed by software producers. Learning objects are small, modular chunks of learning materials and are normally used by software producers to make the production process more efficient. Again, instructional designers can choose to create the product they want from existing learning objects themselves in stead of the producers, as selecting and reusing appropriate learning objects is a fairly straightforward and standardized process that requires no or little technical programming skills.

Despite the increasing availability of support tools, it is still difficult for instructional designers to improve the quality of their design documents, to create programming structures, and to reuse learning objects. Domain specialists and senior instructors often act as instructional designers in the practical field, and are typically inexperienced with production tasks. Therefore, three building-block solutions are proposed in this dissertation. First, a *Developing Design Documents (3D) model* is presented as a three-dimensional decision tool that may help instructional designers to improve their design documents. Second, *instructional software templates* are presented as a promising tool to support instructional designers with the creation of sound instructional software. And third, an *integrative approach* is introduced that may help instructional designers with the reuse of learning objects. In the first solution, instructional designers work in their traditional role but more explicitly tune their design products (blueprints, storyboards, and so forth.) to the production requirements. In the second and third solutions, instructional designers operate in a new production-like role, interrelating the design phase with the production phase by implementing their designs in programming structures and learning objects. Which—combination of—the three solutions can be used depends on factors such as available time of the instructional designers, their capabilities, and suitable support tools (e.g., templates, repositories).

Research Questions

The main research question of this dissertation is if it is possible to support instructional designers to overcome the transition bottleneck by the three proposed building-block solutions. The first research question is if the 3D-model supports the development of design documents that improve software producers' understanding of the design and its translation into technical specifications. The second research question is if instructional software templates support instructional designers in the creation of programming structures that make up didactically sound instructional software. The third research question is if the integrative approach supports instructional designers in the reuse of learning objects that make up didactically sound instructional software.

Overview of the Dissertation

One explorative and three empirical studies are conducted to answer the research questions. Chapter 2 discusses new innovations in the educational field as well as criteria for developing instructional software resulting from these innovations. In this explorative study, theoretical and empirical analyses show that current development approaches for instructional software are not able to satisfy the new criteria of developing instructional software. Lean production is introduced as a promising new development approach that does satisfy all but one criterion. It appears that the criterion of "modeling" is not satisfied due to a lack of design languages that help to transfer design information to the production phase. In order to overcome this problem, and to enable the broader implementation of lean production, design and production should be better interrelated to each other. Three building-block solutions are proposed to accomplish this, focusing the instructional designer's attention on: (1) improving design documents, (2) creating programming structures, and (3) reusing learning objects.

Chapter 3 describes the first empirical study, in which the building-block solution of "improving design documents" is investigated. The lack of organization, detail, and standardization of traditional design documents is discussed. The 3D-model is introduced, which supports instructional designers in creating design documents that are more or less stratified, elaborated, and formalized. The effects of design documents based on the 3D-model are compared with the effects of traditional design documents. The goal of the study is to see if software producers are better able to interpret design documents based on the 3D-model than traditional documents and if this improves their understanding. The degree of agreement between the design documents and scores on technical specification questions is measured, as well as software producers' perceived cognitive load, time investment, and satisfaction.

Chapter 4 describes the second study, in which the building-block solution of "creating programming structures" is investigated. It studies if

software templates can help instructional designers, with little production experience, to create instructional software with an adequate didactical, technical, and authoring quality. Domain specialists who act as instructional designers in their field, with low and high production experience, work with the software templates. The goal of the study is to see if instructional designers with high production experience who work with software templates make products of higher didactical quality and size than instructional designers with low production experience. No differences with regard to technical and authoring quality were expected, as the instructional software templates should level the differences between instructional designers. The volume and quality of the resulting products is measured as well as the satisfaction of the instructional designers. Furthermore, the effects of instructional designers' didactical perspective and development style are explored.

Chapter 5 describes the third study, in which the building-block solution of "reusing learning objects" is investigated. An integrative approach is introduced to improve the reuse of learning objects, with (a) templates, (b) automation, and (c) intermediate product solutions to overcome problems hampering the efficient development of instructional software. Two experiments are described. In the first experiment, domain specialists with low production experience are supported with a combination of the templates and automation solutions: They reuse both small and large learning objects, from both familiar and unfamiliar task domains, to develop instructional software. This experiment is intended to see if domain specialists with little production experience are indeed able to reuse learning objects, and if the type of learning object and the familiarity of the domain does make a difference. The number of reused learning objects, time invested, and instructional designers' opinions were measured. In the second experiment, domain specialists with low production experience are supported by the automation solution in combination with a set of (1) regular templates, (2) extended templates, and (3) intermediate products. The goal of the study is to see what the most effective configuration of these three solutions is. The quality of the resulting products is measured as well as domain specialists' opinions on the provided solutions.

Chapter 6 is the final chapter of the dissertation and presents a general discussion of the reported studies. A review of the main results is given, followed by the main conclusions and practical implications of the studies. Next, the limitations of the conducted studies are discussed. Finally, suggestions for further research are presented and a scenario is described that illustrates how the building-block solutions might work in practice.

References

Jochems, W., van Merriënboer, J. J. G., & Koper, R. (Eds.) (2003). Integrated E-learning: Implications for pedagogy, technology, and organization. London, UK: RoutledgeFalmer.

Merrill, M. D. (2002). First principles of instruction. *Educational Technology, Research and Development*, *50*(3), 43-59.

Reigeluth, C. M. (Ed.) (1999). *Instructional-design theories and models: A new paradigm of instructional theory* (Vol. 2). Mahwah, NJ: Lawrence Erlbaum.

Waters, S. H., & Gibbons, A. S. (2004). Design languages, notation systems, and instructional technology: A case study. *Educational Technology, Research and Development, 52*(2), 57-68.

Woll, C. (2003). *Identifying value in instructional production systems: mapping the value stream.* Unpublished PhD Dissertation, Utah State University, Logan.

Chapter 2

Improving the Development of Instructional Software: Three Building-Block Solutions to Interrelate Design and Production[1]

Abstract

Currently, there is a focus on authentic tasks as the driving force for learning in integrated e-learning systems. This sets new criteria for instructional software, which should become much more flexible and allow for domain modeling and pedagogical modeling. A theoretical analysis and a survey (N = 37) amongst experienced developers show that current development methods are unsuitable to develop such instructional software. New development methods based on lean production promise to satisfy the new criteria, as they emphasize mass-customization by rigorously applying a pull principle throughout the whole development process. However, a potential bottleneck is the lack of design languages to properly transfer the design outcomes to the production phase. Three building-block solutions are proposed to overcome the "transition bottleneck": (1) a Developing Design Documents (3D) model to support designers in stratifying, elaborating, and formalizing design documents; (2) instructional software templates to support designers in producing software independently from producers, and (3) an integrative approach to support designers in reusing learning objects.

[1] Boot, E., van Merriënboer, J.J.G., & Theunissen, N.C.M. (submitted). *Improving the Development of Instructional Software: Three Building-Block Solutions to Interrelate Design and Production*

Current educational, technological, and organizational innovations are rapidly changing the nature of instructional software, and, thereby, the way it is developed. Recent theories of instruction tend to focus on authentic learning tasks that are based on real-life tasks as the driving force for learning (Merrill, 2002; Reigeluth, 1999). The general assumption is that such authentic tasks help learners to integrate the knowledge, skills, and attitudes necessary for effective task performance; give them the opportunity to learn to coordinate constituent skills that make up complex task performance; and eventually enable them to transfer what is learned to their daily life or work settings. This focus on authentic, whole tasks can be found in practical educational approaches, such as project-based education, the case method, problem-based learning, and competency-based learning; in theoretical models, such as Collins, Brown and Newman's (1989) theory of Cognitive Apprenticeship Learning, Jonassen's (1999) theory of Constructive Learning Environments, Nelson's (1999) theory of Collaborative Problem Solving, and Schank's theory of Goal Based Scenario's (Schank, Berman, & MacPerson, 1999); and in instructional design models, such as the Four Component Instructional Design model (Van Merriënboer, 1997).

In addition to educational changes, technological and organizational innovations enable the application of blended learning or integrated e-learning: The combination of face-to-face learning, distance learning, and on-the-job learning. Blended learning is supported by a balanced media-mix of traditional and advanced learning technologies such as books, e-learning, mobile learning, and simulations (Jochems, van Merriënboer, & Koper, 2004). Such integrated e-learning provides both the flexibility to enable the integration of working and learning, in terms of time and place independent learning, and adaptive learning, personalized for individual learners.

The resulting combination of pedagogical considerations (e.g., "How can authentic learning tasks be implemented in the instructional software?"), technological considerations (e.g., "Which media mix is most optimal?"), and organizational considerations (e.g., "How can working and learning be efficiently integrated by means of instructional software?") makes the development process highly complex, requiring a structural approach towards design, production, and implementation.

In this Chapter, we investigate if current development methods provide for such a structural approach to the development of innovative instructional software. First, the new criteria that result from the recent innovations are discussed. Second, a theoretical analysis of current development methods is described. Third, the theoretical analysis is complemented by an empirical analysis, in the form of a survey study. Fourth, lean production is introduced as a new development approach that promises to fit the new situation better. Fifth, the problem of lack of design languages, which hampers the implementation of lean production, is discussed. Sixth, three building-block solutions are proposed to enable the implementation of lean production approaches by supporting designers to (1) improve design documents; (2) use instructional software templates, and (3) reuse learning objects. Finally, conclusions will be drawn

about the consequences of criteria and building blocks for future development of instructional software

New Criteria for Developing Instructional Software

The educational, technological, and organizational innovations lead to four new criteria for instructional software development. The first three criteria are related to the flexibility of development processes and products. For example, a blended learning approach requires the efficient and fast creation of multiple configurations of instructional methods and media ("packages"). Atkinson and Wilson (1969; for more recent discussions, see Gibbons, Nelson, & Richards, 2000; Parrish, 2004) have identified three criteria for developing instructional software related to flexibility. First, *adaptivity,* which is the ability of an instructional software product to adjust itself to learner needs, learner progress, preferences, and choices, provides personalized learning for individual learners. Second, *generativity,* which is the ability to assemble the instructional software product from some combination of parts and sources at the moment of delivery, frees the designer from having to create an infinite variety of products with static designs. Finally, *scalability,* which is the ability to increase the production capacity of instructional software products without a corresponding increase in costs, enables the serving of more and larger target groups.

The fourth, most important criterion is related to the holistic pedagogical view (van Merriënboer & Boot, 2005), central in current educational innovations. A holistic view on learning assumes that complex knowledge and skills are best learnt through cognitive apprenticeship on the part of the learner in a rich environment (Collins, 1988). Experiences are provided for the learners that mimic the apprenticeship programs of adults in trades, or teachers in internship. It is not possible to immerse the learner to the extent that a traditional internship would imply. However, through the use of simulations and meaningful experiences, the learner would learn the ways of knowing of an expert. As a result, the most important problem of a holistic approach is how to deal with complexity.

Most authors introduce some notion of "modeling" to attack this problem. For example, Achtenhagen's (2001) notion of "modeling the model" prescribes a two-step approach to modeling, namely modeling reality and then modeling those models of reality from a pedagogical perspective. For developing instructional software, this implies first the *domain modeling* of realistic tasks and systems in such a way that they are simplified (i.e., reduction of complexity) towards the learner's level of ability while at the same time remaining representative for the "real" world. Second, it implies *pedagogical modeling* of these domain models to facilitate learning, such as the use of modeling examples, coaching, and scaffolding attuned to the expertise, progress, and interests of the learner. This modeling of the model for instructional purposes allows the designer to determine which elements of the

original model can be omitted, and which elements can be increased (not in the original, but introduced for supporting the functions of the model) (Gibbons, Bunderson, Olsen, & Robertson, 1995). For developing instructional materials, a third facet should be added to this modeling process, namely *functional modeling*. This works out the two previous models in order to transfer them by means of design documents from the design phase to the production phase. Functional modeling allows the designer to determine how each element should be presented to the learner. The development criterion of modeling, actually consisting of the three sub-criteria domain, pedagogical, and functional modeling, is conditional for the other three criteria, as adaptivity, generativity, and scalability all depend on an adequate modeling process.

To which degree do current development methods meet the four criteria discussed above? This question could be answered from a theoretical perspective and an empirical perspective, discussed in the next sections.

Theoretical Analysis of Current Development Methods

The vast majority of development methods is based upon the standard Instructional Systems Development (ISD) model, an instantiation of the generic Analysis, Design, Development (also called Production; the technical realization of the design), Implementation, and Evaluation model (ADDIE; Dick & Carey, 1996). Every phase in the ISD model identifies specific types of activities and outcomes, for which different specialists (e.g., designers, producers, visual artists, and so forth) are responsible. Applying the ISD model is typically based upon either a craft production approach or a mass production approach (Woll, 2003). Craft production approaches are directed at providing the highest-quality products, completely adapted towards a specific target group. Development involves highly skilled professionals using flexible, often custom-build tools, in a flexible work process. Products, processes, and tools are not standardized. Developers focus on producing limited quantities: In expanding the volume, costs will rise proportionally. The development of Computer-Based Training (CBT) is a good example of how craft production is applied in the field of instructional software (see, for example, Gibbons & Fairweather, 1998). The first row of Table 2.1 shows that craft production approaches are not able to satisfy any of the new criteria for developing instructional software. Craft production is focused on single solutions for a highly specific target group with particular needs. So design, production, as well as final products, will be focused on that single solution, with very limited use of modeling and adaptivity. There is also no need for products to be modular, so generativity will be difficult to realize. Finally, due to the focus on producing small quantities of unique products, costs will proportionally rise with increase of volume.

Table 2.1
Production Approaches and their Satisfaction of Criteria for Developing Instructional Software

Type of production approach	Criteria			
	Adaptive	Generative	Scalable	Modeling
1. Craft production	No, because it is not necessary for custom-build single-purpose solution	No, because no standards are used and products are monolithic	No, because of the proportional increase in costs of customized processes	No, because modeling is not used as only single-purpose solutions are created
2. Mass production	No, because only single-purpose solutions are created to allow for efficient production	Yes, reached through standardization and modularity of products	Yes, it is an explicit objective, and reached through standardization and modularity of process	No, because modeling is not used as only single-purpose solutions are created
3. Lean Production	Yes, it is an explicit objective and reached through the pull principle	Yes, reached through standardization and modularity of products	Yes, it is an explicit objective, and reached through standardization and modularity of process	No, because the lack of design languages will limit transfer of modeling information

The counterparts of craft production approaches are mass production approaches. These are directed at providing (somewhat) similar products for a broad target group. Development involves narrowly skilled, interchangeable specialists, using expensive, single-purpose tools, in a continuous work process. Products, processes and tools are highly standardized and modularized. Developers focus on producing large quantities: With every increase in volume, costs per unit will decrease. The development of e-learning materials is a good example of how mass production is applied in the field of instructional software. For example, recent standardization efforts (see Collis & Strijker, 2004, for

examples in the military, commercial, and academic fields) explicitly refer to standardized, modularized approaches to increase scalability. The second row of Table 2.1 shows that mass production approaches are not able to satisfy the criteria of modeling and adaptivity for the same reasons as craft production models. However, the criterion of generativity can be satisfied as resulting products are highly standardized and modular. Also, scalability is an explicit objective of mass production.

A third development approach, lean production (third row of Table 2.1), may better meet the new criteria for developing instructional software. Before we discuss this alternative, however, the empirical analysis of the craft and mass production approaches will be presented.

Empirical Analysis of Current Development Methods

To investigate the practical application of current development methods and the degree to which they meet the criteria, a survey study has been conducted. In this study, a questionnaire was used to gather opinions of experienced developers of instructional software on their current practices and experienced problems. A special focus is on the transition of information between the design phase and the production phase (the "transition bottleneck"), as defined by the third modeling step, namely, functional modeling.

Method

Respondents. Thirty-seven developers of instructional software from the United States of America (n = 17) and the Netherlands (n = 18), working in large academic, commercial, and military organizations, participated in the study. All participants were male and their age varied between 24 and 58 years.

Materials. An on-line questionnaire was used to gather information on the respondents and the problems they experienced in the development process. First, to investigate the respondents' background, they were asked to indicate their overall experience, and the different roles they fulfilled, relevant to developing instructional software. They were also asked whether or not they applied structural, phased approaches based upon the ISD model, and who the responsible persons were for creating the design documents as output of the design phase. Second, to investigate possible development problems, the respondents were asked to rate on a 5-point scale for each of the five ADDIE phases the occurrence of seven typical development problems. Third, the respondents were asked to rate on a 5-point scale 21 statements that focused on possible causes and consequences of problems in transferring information from the design phase to the production phase (see Table 2.4; note that questions 7 to 21 were only presented to the USA respondents). The 21 statements were established on the basis of suggestions from experienced developers. Fourth, to investigate the need for new solutions for the design-production transition bottleneck, the respondents were asked to indicate on a 5-point scale whether

they needed solutions for the transition bottleneck for either themselves or for their organization (questions 22 and 23 of Table 2.4).

Data analysis. A factor analysis (Principle Component analysis with Varimax rotation) was used to identify the main development problems from the scores of the seven problems for each of the five ADDIE phases (35 items). Using a factor analysis for data reduction, a small number of issues could be identified that explains most of the variance observed in the larger number of item scores. Next, to determine the intra-item reliability of items within each factor, Cronbach's alpha is computed. In general, an alpha larger than .70 is regarded as satisfactory for drawing conclusions about different groups. Scale scores for each factor were obtained by adding item scores within the scale, and transforming crude scale scores linearly to a 0-100 range, with higher scores indicating more problems. Differences between groups (nationality, role, or type of organization) with respect to the issues were tested by MANOVA.

With regard to the possible causes and consequences of the transition bottleneck, and the perceived need for new solutions, one-sample *T*-tests were used to test for differences between the ratings and the neutral score of 3.

Results

First, the number of years of experience in a particular role was used to determine the main specialization of the respondents. The respondents' mean experience with developing instructional software was 16.97 years (*SD* = 12.69). Their main specialization was designer, with a mean experience of 6.11 years (*SD* = 5.39). Furthermore, they were experienced as project leader (*M* = 4.97 years, *SD* = 4.42), manager or policy-maker (*M* = 2.03 years, *SD* = 3.84), multi-media specialist (*M* = 3.05 years, *SD* = 4.55), or programmer (*M* = 0.81 years, *SD* = 3.09). All respondents indicated that they used structural, phased approaches based upon the ISD model. The respondents indicated that in their organization the following persons were responsible for creating training blueprints: In 16 cases, only the designers were responsible; in 6 cases, only producers, and in 15 cases, combinations of designers and producers. So, in most cases, designers were either responsible for or directly involved in the creation of design documents.

Second, with respect to possible development problems, the factor analysis identified four factors (see Table 2.2). Combined, they explained 53% of the total variance.

Table 2.2
Factor Loadings of the 35 Items covering Five ADDIE Phases in Combinations with Seven Development Problems

Item	Scale 1	Scale 2	Scale 3	Scale 4
Internal design and development difficulties				
Problems with managing the design phase	.82			
Problems with production activities in the production phase	.67			
Problems with production phase too time-consuming	.65			
Problems with design phase too time-consuming	.64			
Problems with design activities in the design phase	.61			
Problems with managing the analysis phase	.56			
Problems with implementation phase too time-consuming	.56			
Problems with analysis phase too time-consuming	.53			
Problems with managing the evaluation phase	.51			
Problems with limited return on investment in the design phase	.49			
Problems with analyzing in analysis phase	.46			
Problems with managing the production phase	.41			
Rolling-out difficulties				
Problems with evaluation too labor intensive		.69		
Problems with cooperation with stakeholders in evaluation phase		.66		
Problems with changing requirements and conditions in implementation phase		.65		
Problems with stakeholders in implementation phase		.62		
Problems with limited return on investment in implementation phase		.61		
Problems with changing requirements and conditions in evaluation phase		.61		
Problems in evaluation phase with input from previous phase		.60		
Problems with managing the implementation phase		.58		

Problems in implementation phase with input	.53		
Problems with implementation activities in implementation phase	.48		
Problems with limited return on investment in evaluation phase	.48		
External design and development difficulties			
Problems with changing requirements and conditions in production phase		.73	
Problems in production phase with input		.69	
Problems with cooperation with stakeholders in production phase		.67	
Problems with cooperation with stakeholders in design phase		.64	
Problems with evaluation in evaluation phase		.61	
Problems with changing requirements and conditions in design phase		.59	
Problems in design phase with input		.49	
Problems with limited return on investments in production phase		.40	
Front-end analysis difficulties			
Problems with changing requirements and conditions in analysis phase			.84
Problems in analysis phase with input			.68
Problems with limited return on investments in analysis phase			.59
Problems with cooperation with stakeholders in analysis phase			.52

The first factor can be interpreted as *internal design and production difficulties*: Respondents experience problems in organizing and executing (i.e., designing, producing) the design and production phases. The second issue can be interpreted as *rolling-out difficulties*: Respondents experience difficulties with executing the implementation and evaluation phases. The third factor can be interpreted as *external design and production difficulties*: Respondents experience problems in dealing with external conditions such as quality of input, cooperation between stakeholders, and changing requirements and conditions of the design and production phases. The fourth factor can be interpreted as *front-end analysis difficulties*: Respondents experience problems with executing the analysis phase. Table 2.3 presents the number of associated items with a particular factor, and Cronbachs' Alpha for the items contributing to that factor. MANOVA's showed no significant differences for nationality, role, or type of organization on any of the factors.

Table 2.3
Number of Items, Cronbach Alpha's, and Percentages of Explained Variance for the Four Factors

Factors	Number of items	Cronbach's Alpha	Explained Variance
Internal design and production difficulties	12	.85	16 %
Rolling-out difficulties	10	.85	15 %
External design and production difficulties	8	.84	12 %
Front-end analysis difficulties	5	.67	9 %

Finally, Table 2.4 shows the ratings on possible causes and consequences of the transition bottleneck, as well as the need for new solutions for that bottleneck.

Table 2.4
Ratings on Causes, Consequences, and Need for New Solutions for the Transition Bottleneck

Questions	*M*	*SD*
Causes of problems in transferring information from design to production		
1. Lack of instructional design information	2.60	1.22
2. Lack of modularization	2.76	1.25
3. Lack of clear design languages	2.80	1.21
4. Lack of structure in design languages	2.50	1.22
5. Too much information transferred in design documents	2.13*	.86
6. Too little information transferred in design documents	2.70	1.05
7. Producers' lack of design knowledge	2.65	1.22
8. Producers making design decisions	2.70	1.26
9. Designers' lack of production knowledge	3.00	1.00
10. Designers making production decisions	2.82	1.07
11. Communication between designers and producers starts too late	3.00	1.32
12. Communication between designers and producers starts too early	1.94**	0.83
13. Cooperation between designers and producers starts too late	2.82	1.27
14. Cooperation between designers and producers starts too early	2.11**	0.78
15. Instructional Design models incomplete	2.76	1.14
16. Instructional Design models providing too little guidance	2.71	1.10
Consequences of problems in transferring information from design to production		
17. Too long development process	3.47	1.33
18. Planning of development process difficult	3.41	1.12
19. Structuring development teams difficult	3.00	1.22
20. Unpredictability of character of end product	2.76	1.09
21. Unpredictability of pedagogical quality of end product	2.65	1.27
Need for new solutions		
22. For the transition bottleneck for myself	2.88	1.26
23. For the transition bottleneck for my organization	3.06	1.34

All questions scored on a 5-point Likert-scale ranging from 1 ("totally disagree") to 5 ("totally agree")
* $p < .05$
** $p < .01$

First, respondents rated the possible cause "too much information transferred in design documents" as less relevant than the neutral score (M = 2.13, SD = .86; t = -3.33, p < .05). Second, they rated the possible cause "communication between designers and producers starts too early" as less relevant than neutral (M = 1.94, SD = .83; t = -5.29, p < .01). Third, respondents rated the possible cause "cooperation between designers and producers starts too early" as less relevant than neutral (M = 2.11, SD = .78; t = -4.65, p < .01). There were no significant differences between nationality, organization, and role on the ratings for possible causes and consequences and the need for new solutions.

Discussion

It appears that developers experience problems in organizing, managing and executing each phase, particularly in the design and production phases as indicated by the factors "internal design and production difficulties" and "external design and production difficulties", together explaining 28% of the variance. However, developers were not able to indicate clear causes of these problems. On the contrary, they indicated for three issues that they are *not* the cause of the identified problems, namely, "too much information transferred," "too early communication," and "too early cooperation." Furthermore, they did not indicate possible consequences of the problems. Finally, they did not indicate a need for new solutions for themselves or for their organizations. It seems that designers report problems from a vague feeling rather than from experiencing concrete bottlenecks. A possible explanation is that they are educated in, and experienced with ISD-based development methods, and are not familiar with alternative methods that could help to overcome the transition bottleneck. Or, as Womack, Jones, and Roos (1990) stated, workers in a particular manufacturing model will not criticize this model nor move to another model unless there is a real crisis. The absence of a need for new solutions also implies that developers will probably be rather skeptical in implementing new improvements.

A limitation of this study is that the respondents predominately had design experience and less production experience. This could possibly explain why they were not able to indicate clear causes and consequences of development problems. Producers, for example, could have been better able to indicate what they would need to improve the development process. A second limitation concerns the modest number of respondents, which limits the results of the factor analysis. Setting aside these limitations, the results of our empirical analysis are fully in line with the theoretical analysis.

Lean Production as an Alternative Development Approach

The theoretical and empirical analyses show that development methods based upon current craft and mass production approaches do not satisfy the new criteria for developing instructional software. The practical application of these production approaches is not without problems either. In the

manufacturing industry, lean production is introduced as a new approach for development methods, to overcome the problems of craft and mass production (Womack et al., 1990). Lean production aims to provide a high variability of high quality products, which are flexibly to adapt to different clients. Development according to the lean production approach involves autonomous, multi-skilled expert-teams, using *flexible* automated tools in a standardized and modularized work process. Lean production raises efficiency through the continuous, incremental improvement of work processes.

One important implication of lean production is the radical application of the "pull principle." Craft- and mass-production approaches are supply-oriented: Developers "push" the product they think is appropriate for the client forward through the work process. Lean production, however, is demand-oriented. In the field of Instructional Systems Development (ISD), this implies that developers have to consider the specific needs of clients and create their products accordingly. The pull principle applies to both intermediate products, transferred between the phases, and final products. Figure 2.1 compares the supply-oriented model with the demand-oriented model, indicating that according to the pull principle, designers pull information from the analysts, producers pull information from the designers, implementers pull information from the producers, and evaluators pull information from the implementers. This demand-oriented principle ensures a more effective transition process of information and products.

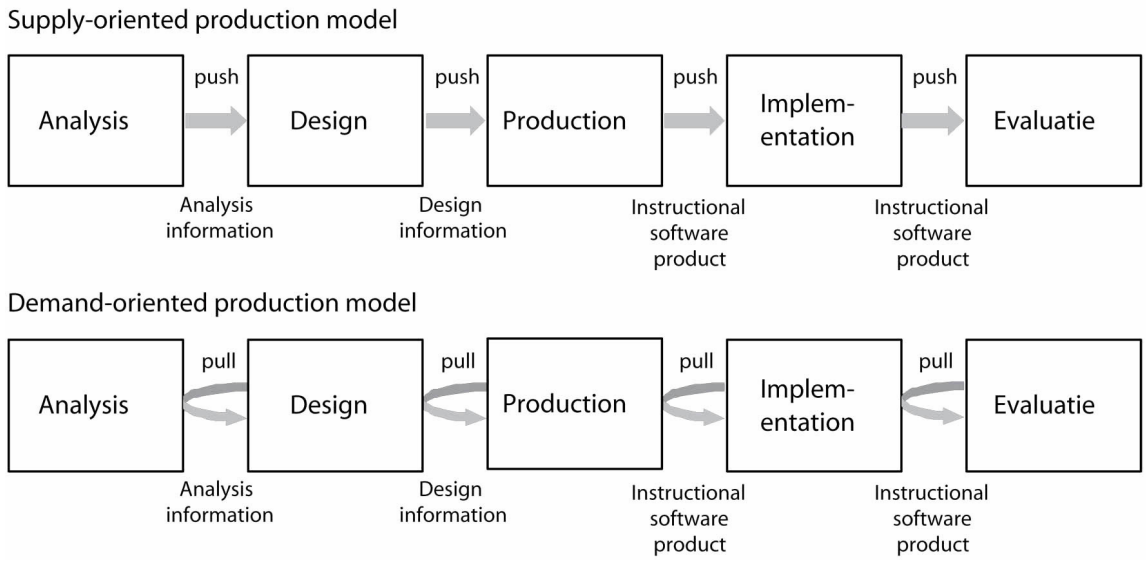

Figure 2.1. Supply versus demand oriented Instructional Systems Development (ISD) models.

Another important implication of lean production is "waste-reduction," which is the continuous process of measuring and analyzing the development

process and (intermediate) products to improve quality, increase standardization, and limit waste (Ohno, 1988). Waste is defined as unnecessary delays, defects, and redundancies in the development process. This puts much emphasis on the quality of the transition of information or products, because each transition must be optimal the first time. Otherwise, time-consuming iterations are necessary to provide additional explanations or correct errors.

With respect to the criteria presented in our theoretical analysis of current development methods, the lean production approach is the only approach that meets the criterion of adaptivity (see Table 2.1). Therefore, lean production is suitable for the development of innovative instructional software (Woll, 2003). Schellekens (2004) suggests a similar, process-focused strategy, which promotes "mass-customization" with a high degree of adaptation to the needs of clients, high design quality, and volume flexibility.

As can be seen in the third row of Table 2.1, satisfying the criterion of modeling, particularly the step of functional modeling, remains difficult in the lean production approach. The application of the pull principle between the design and development phases implies that developers have to pull the information they require from designers. However, the designers lack the necessary means to provide the producers with information that ensures an unequivocal interpretation by producers. They have a different background than producers (educational vs. technological) and use different tools (analysis and design tools vs. technical production tools). For applying the pull principle between other phases, the signaled problem is less relevant because analysts and designers have the same background, and because developers and implementers, as well as implementers and evaluators, exchange concrete products.

Lack of Common Design Languages

The lack of standardized design languages, familiar to both designers and producers, is a fundamental problem in the transition between design and production. Such design languages should be able to allow for functional modeling by capturing and describing the domain and pedagogical models at a level of detail ensuring that different producers interpret them unequivocally (Waters & Gibbons, 2004). Design languages require notation systems to convey their message by means of symbolic, graphical, textual or other conventions. An example of a graphical modeling language, not bound to the field of instructional software development, is the Unified Modeling Language (UML; Booch, 1994). The notation system of UML (i.e., diagrams) enables both designers and producers to describe and understand a design. Recent attempts to introduce design languages in the field of instructional software development are IMS Learning Design (IMS LD; Koper & Tattersall, 2005) and the Educational Environment Modeling Language (E2ML; Botturi, in press). Both languages are promising but not yet able to provide a complete solution for the transition bottleneck. IMS LD is limited to the configuration of a pedagogical model in an

IMS LD compatible e-learning system. E2ML is limited to describing instructional *design* issues such as learning goals, roles, actions, and resources, instead of (relating this to) instructional *software* issues such as its interface design, interaction design, and information flow.

As long as there are no complete design languages available in the field of instructional software development, iteration as proposed by agile methods (e.g., see http://www.agilealliance.com) might possibly provide a solution to overcome the transition bottleneck. Iteration implies that designers and producers model and produce the instructional software incrementally and in direct contact with each other, rather than relying on the once-only transfer of formalized information (i.e. functional modeling) between the design and production phases. Iteration is popular in the fields of software engineering (Fowler, 2004) and instructional development (e.g., Reigeluth & Nelson, 1997; Tennyson, 1995). However, three problems limit the value of iteration: (a) Lack of expertise of designers and producers, (b) outsourcing of production, and (c) lower efficiency.

The first limitation of iteration is that one of the characteristics of instructional software development is the participation of domain specialists such as subject matter experts and instructors, with relatively less instructional design and software production expertise (Hoogveld, Paas, Jochems, & van Merriënboer, 2003; Spector & Muraida, 1997). Iteration, however, requires considerable expertise in order to determine exactly when and how iteration should take place (Verstegen, 2003). This problem may be solved by Verstegen's methodology for the development of a needs statement. In this method, directed at establishing a thorough needs assessment before starting the actual development process, a series of workshops is organized with stakeholders such as clients, teachers, learners, and instructional designers. Under guidance of an experienced discussion leader, they proceed iteratively through all ISD phases in a structured and standardized manner, and record their assumptions and (provisional) decisions with regard to design, production, implementation, and evaluation issues. The standardized method and structured discussion overcome the problem of lack of expertise. Also, if producers are involved in the workshops, the resulting needs-assessment documents may provide development information that is understood and accepted by both designers and producers, thereby avoiding the need to rely solely on formal transfers of design documents.

However, the workshop methodology will often be impossible due to the second limitation of iteration, namely outsourcing of production. In large organizations and in professional development projects, there is often a strict juridical and organizational separation between design and production due to outsourcing of production activities to external companies. Note that production-related input in Verstegen's (2003) method can be accomplished by involving other producers, to promote at least understanding of the development information by the ultimate producers.

The third limitation of iteration is that it may reduce efficiency because it costs extra time, and there is no guarantee that this will be regained in a later phase of the development process. For example, Verstegen's (2003) method explicitly emphasizes iteration within the needs-assessment phase to prevent iteration between later development phases.
Summarizing, development methods based on lean production promise to satisfy the new criteria for instructional software development, except domain, pedagogic, and—in particular--functional modeling, due to the lack of common design languages and the limitations of iteration. A possible solution should focus on supporting designers to provide producers with exactly the functional modeling information they need.

Three Building-Block Solutions

Currently, production-related information is typically embedded in three types of building blocks for the production process: (a) design documents as input, (b) programming structures as throughput, and (c) learning materials as output (see Figure 2.2).

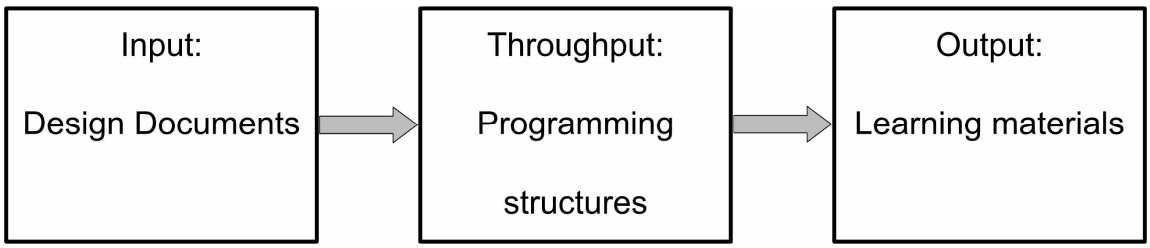

Figure 2.2. The three artifacts in the development process that embed production-related information.

In order to better interrelate the design phase into the production phase, it is proposed that designers' attention be focused on these three building blocks, thereby preventing the need to rely solely on design documents and/or iteration. Three building-block solutions are proposed to compensate for the limited production expertise of the typical designer: First, the Developing Design Documents (3D) model to support designers to improve design documents; second, instructional software templates to support designers to create programming structures, and third, an integrative approach to support designers to reuse learning objects. According to the first solution, designers work in their traditional role but use functional modeling to interrelate their design more explicitly to production. According to the second and third solutions, designers operate in a producer's role, using programming structures

and learning objects to interrelate their designs more explicitly to production. The next sections discuss the three solutions.

The 3D-model

In functional modeling, design documents such as training blueprints and storyboards serve as input for creating technical specifications by producers. Design documents may be difficult to interpret for three reasons: (a) different instructional and technical structures are often not meaningfully organized; (b) different levels of detail are mixed up, and (c) different expressions are used in a non-standardized manner. With regard to meaningful organization, Gibbons' model of Design Layers (Gibbons, 2003) may be used for *stratification* of the instructional software design on seven, interrelated layers: Content, strategy, control, message, representation, media logic, and data management. Each layer is typified by the designer's selection of design languages pertaining to the solution of different instructional design sub problems. Together, the functional designs at each layer make up the total design. Stratification helps to determine the relations between the functionally different instructional and technical structures, while at the same time staying cognizant of the need for integration of those structures within the complete design.

With regard to mixing up different levels of detail, the three perspectives of Fowler (2004) may be used for the *elaboration* of the instructional software design: (a) A conceptual perspective, with more or less superficial and descriptive information; (b) a specification perspective, with more or less comprehensive and detailed information, and (c) an implementation perspective, with more or less technical and meticulous information. Elaboration helps to determine the required level of detail, depending on the capabilities of the designer and the needs of the producer.

With regard to the use of non-standardized expressions, designers may reach *formalization* of their design by making their informal and formal design languages explicit. They should strive for (combinations of) formal languages, but depending on their capabilities and the needs of the producer, they can also select (combinations of) informal languages. Formalization helps to determine the required level of standardization.

The 3D-model uses stratification, elaboration and formalization as its three dimensions. Designers, with or without producers, may first analyze their design situation in order to determine the optimal configuration of the 3D-model (e.g., What kind of designers and producers are involved? What kind of training is the design made for? Which support tools are available?), and then use this configuration to stratify, elaborate, and formalize their design documents. Figure 2.3 presents the 3D-model in its full configuration, in which all dimensions are completely utilized. The 3D-model provides producers with

insight in the underlying structure and content of the functional model, even when the design languages used are deficient.

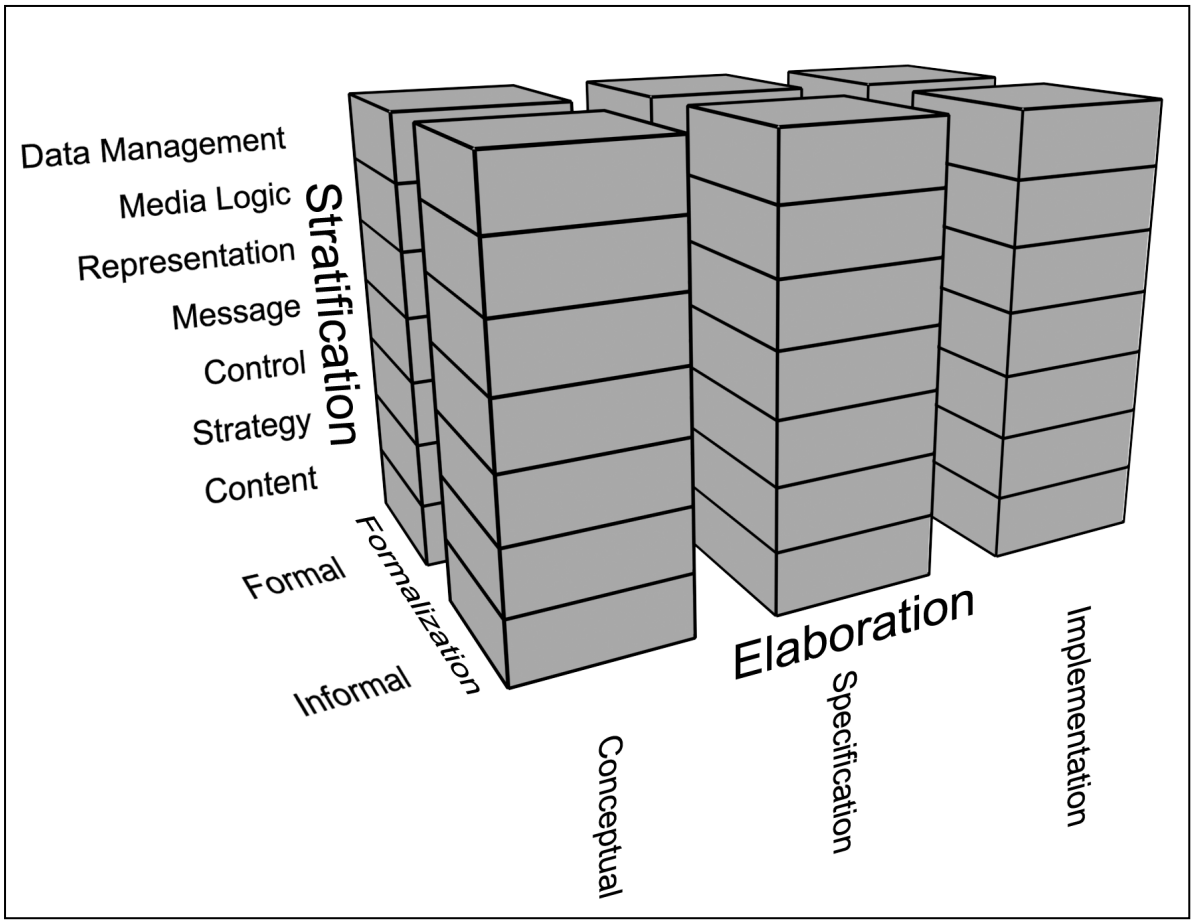

Figure 2.3. The Developing Design Documents (3D) model.

Instructional Software Templates

E-learning systems and authoring tools often provide instructional software templates, which producers can use to easily create or adapt programming structures that make up the instructional software. This "programmer-less-authoring" (Hedberg & Sims, 2001) is based on automation of routine tasks and intuitive interfaces such as "wizards." They provide support on three levels. First, on the authoring level, the templates offer prefabricated "moulds" of programming structures to implement the lessons, practice items, test questions, examples, cases, feedback, learner support, and so forth into the instructional software. Second, on the technical level, the templates automatically produce programming structures that are compliant with current e-learning technologies, learning technology standards, and different operating systems. Third, on the pedagogical level, the templates provide prefabricated structures, for example, drill-and-practice, concept learning, mastery learning,

and case-based learning. On the one hand, producers use templates to speed up their authoring and technical activities and to receive pedagogical support for creating programming structures without support from instructional designers. On the other hand, those designers may use templates to speed up their pedagogical activities and receive authoring and technical support.

Instructional software templates explicitly interrelate design and production. First, designers can choose not to rely on the producers at all and select and instantiate appropriate templates without the involvement of producers. This way, they are able to create and assemble the programming structures they want, thereby making their own instructional software. Second, designers can provide producers with design information by means of implementing their design principles. These templates will then force producers to apply particular pedagogical principles in the instructional software. Third, designers might provide producers with design information by means of example products they have created with templates. Optionally, these examples can be presented to the producers together with the design documents.

Learning Objects

To increase the efficiency of design and production, there is currently much emphasis on the *reuse* of learning materials. If learning materials are divided into small, modular chunks, often called "learning objects," developers will be able to combine and recombine those objects to create new learning materials. Van Merriënboer and Boot (2004) identify six problems with the current reuse of learning objects. The first three problems relate to the nature of learning objects: (1) The *metadata problem* refers to the fact that it is difficult and extremely labor-intensive to specify metadata for large sets of learning objects; (2) the *arrangement problem* refers to the fact that combining and sequencing learning objects into larger arrangements is not always easy and self-evident, and (3) the *exchange problem* refers to the fact that it may be difficult from a psychological viewpoint (e.g., due to the "not-invented-here" syndrome) or organizational viewpoint (e.g., due to security issues or intellectual property rights) to exchange learning objects between developers and between e-learning systems. The remaining three problems arise because current approaches of reuse are not consistent with the holistic pedagogical view: (4) The *context problem* refers to the fact that effective learning objects cannot be created in isolation without an implicit or explicit instructional setting, target group, and other contextual descriptors; (5) the *pedagogical function* problem refers to the fact that it is difficult to express pedagogical intentions for a learning object by means of technical properties such as metadata, and (6) the *correspondence problem* refers to the fact that a developer working from a holistic viewpoint will typically not search for one particular learning object but rather for a *set* of meaningfully interrelated learning objects that is aimed at the construction of *one* rich cognitive representation.

Van Merriënboer and Boot (2004) propose an integrative approach, stressing four solutions to improve the reuse of learning objects. The first

solution is to reedit instead of reuse learning objects. This increases the chance that the developer will find a useful learning object because it becomes less important to find exactly what is needed. The second solution is to use templates instead of instantiations as learning objects. Templates allow for the easy modification of learning objects (e.g., a change of an American grading system to a European grading system), making them useful for a broader range of situations. The third solution is to automate the creation and reuse of learning objects. The use of automatic analysis of multimedia content and the semantic indexing of this content in metadata fields makes reuse more cost-effective and also yields more objective metadata than indexing by hand. The final solution is to use intermediate products in addition to final products as learning objects. Intermediate products, such as task analysis results and lesson plans, contain rich information that describes the final products for which they were made. This rich information is more suitable than metadata to provide input for searching suitable learning objects.

The integrative approach to the reuse of learning objects explicitly supports designers to interrelate design to production. First, designers can choose not to rely on producers at all and independently select and reuse appropriate learning objects to assemble the instructional software product they want. Second, designers can provide producers with design documents illustrated with example sets of learning objects they have assembled.

Conclusions and Discussion

New criteria for instructional software development are set by recent pedagogical, technological and organizational innovations: Adaptivity, generativity, scalability, and last but not least, modeling. From theoretical and empirical analyses, which clearly corroborate each other, it appears that existing instructional software development methods based on a push-principle do not satisfy all criteria. Lean production, based upon the pull-principle, is suggested as a new development approach to enable the required mass-customization of instructional software. However, lean production also suffers from the fundamental problem of a lack of design languages to transfer information from the design phase to the production phase. In order to overcome this problem, designers may use production building blocks that prevent sole reliance on design languages and/or iteration. We proposed three building block solutions to support designers in functional modeling: The 3D-model to improve design documents, instructional software templates to create programming structures, and the integrative approach for reusing learning objects.

The suggested building-block methods are predominantly based on practical experiences and theoretical as well as empirical analyses. Further research might go in three directions. First, it should validate the actual value of the building-block solutions, separately and in combination, on the success of the transition between design and production. In particular, it is interesting to study the application of the three solutions by domain specialists such as subject

matter experts and teachers, because although they are inexperienced in instructional design and software production, they are often the persons involved in actual instructional software development projects. Second, as our empirical analysis shows, designers are likely to be rather skeptical towards new solutions. Changing the instructional software development process as drastically as lean production approaches suggest, and also introducing the proposed building-block solutions that require other design skills than before, will probably meet resistance from designers. So, further research and validation should also be aimed at the development of innovation models that help to promote acceptance of new solutions by designers. Third, further research may pertain to the roles of designers and producers. The pull principle suggests that designers should be fully responsible for solving the transition bottleneck, as producers are the "demanding party." This does not imply that designers should provide any solution that producers demand. Further research should be aimed at clarification of new roles for producers and designers.

This study has some clear practical implications for the use of the three building-block solutions, either alone or in combination. First, they allow designers to improve their design documents through the analysis of instructional software templates and learning objects used by a production team. This yields useful product information and informs designers about the capabilities and preferences of the producers. Second, they allow designers to improve their instructional software templates through the analysis of multiple sets of learning objects and design documents. This yields useful design information to serve as input for creating new templates. Third, they allow designers to improve their reuse of learning objects, as the integrative approach incorporates both using design documents (called "intermediate products") and using instructional software templates. This way the three proposed solutions may offer a first step toward the implementation of the holistic pedagogical view, with a focus on authentic learning tasks, in innovative instructional software.

References

Achtenhagen, F. (2001). Criteria for the development of complex teaching-learning environments. *Instructional Science*, *29*, 361-380.

Atkinson, R. C., & Wilson, H. A. (1969). *Computer assisted instruction: A book of readings.* New York: Academic Press.

Booch, G. (1994). *Object-oriented analysis and design with applications*. Redwood City, CA: Benjamin/Cummings.

Botturi, L. (in press). E2ML: A visual language for the design of instruction. *Educational Technology, Research and Development.*

Collins, A. (1988). *Cognitive apprenticeship and instructional technology* (Tech. Rep. No. 6899). Cambridge, MA: BBN Labs Inc.

Collins, A., Brown, J. S., & Newman, S. E. (1989). Cognitive apprenticeship: Teaching the craft of reading, writing, and mathematics. In L. B. Resnick (Ed.), *Knowing, learning, and instruction: Essays in honor of Robert Glaser* (pp. 453-493). Hillsdale, NJ: Lawrence Erlbaum.

Dick, W., & Carey, L. (1996). *The systematic design of instruction* (4th Ed.). New York: Harper Collins.

Fowler, M. (2003). *UML distilled: A brief guide to the standard object modeling language.* Boston, MA: Addison-Wesley.

Gibbons, A. S. (2003). What and how do designers design? A theory of design structure. *Tech Trends, 47*(5), 22-27.

Gibbons, A. S., & Fairweather, P. G. (1998). *Computer-based instruction: Design and development.* Englewood Cliffs, NJ: Educational Technology Publications.

Gibbons, A. S., Bunderson, C. V., Olsen, J. B., & Robertson, J. (1995). Work models: Still beyond instructional objectives. *Machine-Mediated Learning, 5*(3&4), 221-236.

Gibbons, A. S., Nelson, J., & Richards, R. (2000). The nature and origin of instructional objects. In D. A. Wiley (Ed.), *The instructional use of learning objects* (pp. 25-58). Bloomington, IN: AECT.

Hedberg, J., & Sims, R. (2001). Speculations on design team interactions. *Journal of International Learning Research, 12*(2/3), 193-208.

Hoogveld, A. W. M., Paas, F., Jochems, W. M. G., & van Merriënboer, J. J. G. (2002). Exploring teachers' instructional design practices: Implications for improving teacher training. *Instructional Science, 30*, 291-305.

Jochems, W., van Merriënboer, J. J. G., & Koper, R. (Eds.) (2003). *Integrated E-learning: Implications for pedagogy, technology, and organization*. London, UK: RoutledgeFalmer.

Jonassen, D. H. (1999). Designing constructivist learning environments. In C. M. Reigeluth (Ed.), *Instructional design theories and models: A new paradigm of instructional theory* (Vol. II) (pp. 371-396). Mahwah, NJ: Lawrence Erlbaum.

Koper, R., & Tattersall, C. (Eds.) (2005). *Learning design: A handbook on modeling and delivering networked education and training.* Heidelberg, Germany: Springer-Verlag.

Merrill, M. D. (2002). First principles of instruction. *Educational Technology, Research and Development*, *50*(3), 43-59.

Nelson, L. M. (1999). Collaborative problem solving. In C.M. Reigeluth (Ed.), *Instructional design theories and models: A new paradigm of instructional theory* (Vol. II) (pp. 241-267). Mahwah, NJ: Lawrence Erlbaum.

Ohno, T. (1988). *Toyota production system: Beyond large-scale production*. Portland, OR: Productivity Press.

Parrish, P. E. (2004). The trouble with learning objects. *Educational Technology, Research and Development, 52*(1), 49-67.

Reigeluth, C. M. (Ed.) (1999). *Instructional-design theories and models: A new paradigm of instructional theory* (Vol. 2). Mahwah, NJ: Lawrence Erlbaum.

Reigeluth, C. M., & Nelson, L. M. (1997). A new paradigm of ISD? In R. M. Branch, B. B. Minor, & D. P. Ely (Eds.), *Educational media and technology yearbook* (Vol. 22, pp. 24-35). Englewood, CO: Libraries Unlimited.

Schank, R. C., Berman, T. R., & MacPerson, K. A. (1999). Learning by doing. In C.M. Reigeluth (Ed.), *Instructional design theories and models: A new paradigm of instructional theory* (Vol. II) (pp. 161-181). Mahwah, NJ: Lawrence Erlbaum.

Schellekens, A. (2004). *Towards flexible programmes in higher professional education.* Unpublished PhD dissertation. Open University of the Netherlands, Heerlen.

Schwartz, B. (2004). *The paradox of choice: Why more is less.* New York: Harper Collins Publishers.

Spector, J., & Muraida, D. (1997). Automating design instruction. In S. Dijkstra, N. Seel, F. Schott, & D. Tennyson (Eds.). *Instructional Design: International Perspectives* (Vol. 2) (pp. 59-81). Mahwah, NJ: Lawrence Erlbaum.

Collis, B., & Strijker, A. (2004). Technology and human issues in reusing learning objects. *Journal of Interactive Media in Education, 2004* (4). Special Issue on the Educational Semantic Web.

Tennyson, R. D. (1995). Four generations of Instructional System Development. *Journal of Structural Learning, 12*(3), 149-164.

Van Berlo, M. P. W. (2005). *Instructional design for team training: Development and validation of guidelines.* Unpublished PhD dissertation. Catholic University of Leuven, Belgium.

Van Merriënboer, J. J. G. (1997). *Training complex cognitive skills*. Englewood Cliffs, NJ: Educational Technology Publications.

Van Merriënboer, J. J. G., & Boot, E. W. (2005). A holistic pedagogical view of learning objects. In J. M. Spector, S. Ohrazda, P. van Schaaik, & D. A. Wiley (Eds.), *Innovations in instructional technology: Essays in honor of M. David Merrill* (pp. 43-64). Mahwah, NJ: Lawrence Erlbaum.

Verstegen, D. M. L. (2003). *Iteration in instructional design: An empirical study on the specification of training simulators.* Unpublished PhD dissertation. Utrecht University, Utrecht, The Netherlands.

Waters, S. H., & Gibbons, A. S. (2004), Design languages, notation systems, and instructional technology: A case study. *Educational Technology, Research and Development, 52*(2), 57-68.

Woll, C. (2003). *Identifying value in instructional production systems: mapping the value stream.* Unpublished PhD dissertation. Utah State University, Logan.

Womack, J. P., Jones, D. T., & Roos, D. (1990). *The machine that changed the world.* New York: HarperCollins.

Chapter 3

Stratification, Elaboration, and Formalization of Design Documents: Effects on the Production of Instructional Materials[2]

Abstract

Designers and producers of instructional materials lack a common design language. As a result, producers have difficulties translating design documents into technical specifications. The Developing Design Documents (3D) model is introduced to improve the stratification, elaboration, and formalization of design documents. It is hypothesized that producers working with improved documents (n = 8) show a more efficient translation process and more satisfaction with the design documents than producers working with traditional documents (n = 8). As expected, in the improved documents group, a higher agreement was found between the design documents and the technical specifications, which also required less time and less perceived cognitive load for their production. There were no differences on satisfaction with the design documents. The study shows that designers, working with the 3D-model, are able to improve design documents, resulting in a better translation process and integration of design and production.

[2] Boot, E., Nelson, J., van Merriënboer, J. J. G., & Gibbons, A. S. (submitted). *Stratification, Elaboration, and Formalization of Design Documents: Effects on the Production of Instructional Materials*

Developing instructional software is becoming increasingly complex. An important reason is the increasing interest in competency-based learning, which is characterized by learning integrated sets of knowledge elements, skills, and attitudes to recognize and solve problems in a variety of real-life situations. Training programs for competency-based learning are often problem, project, or case based and typically include authentic, realistic learning tasks as the driving force for learning (cf. Merrill, 2002). Instructional software for such training programs often has the character of *blended learning* or *integrated e-learning* (Jochems, van Merriënboer, & Koper, 2004), emphasizing flexible and adaptive learning paths. Developing such complex instructional software predominantly takes place in large projects, conducted by multidisciplinary teams, and based upon modular, object-oriented approaches towards specification and production.

The basic development model is the Instructional Systems Development (ISD) model, an instantiation of the generic Analysis, Design, Development, Implementation, and Evaluation model (ADDIE; Dick & Carey, 1996). Every phase in the ISD model identifies specific types of activities and outcomes, for which different specialists (e.g., designers, producers, visual artists and so forth) are responsible. The outcomes of a preceding phase are mostly transferred to the next phase by means of *design documents*. In the design phase, for instance, instructional designers create a design based upon information from the preceding analysis phase, consisting of a learning hierarchy, target group analysis results, a context description, and so forth. In the case of competency-based learning, the design is likely to be based on models such as the four-component instructional design model (4C/ID model; van Merriënboer, 1997) and described in a particular design document, in this phase often called a *training blueprint* (van Merriënboer, Clark, & de Croock, 2002). In addition, designers can provide guidelines with respect to the implementation of the training blueprint in the instructional software, for instance by means of *storyboards* (e.g., Driscoll, 1998). In the subsequent development phase, producers such as systems integrators, multimedia specialists and programmers, interpret, elaborate and transform the training blueprints and storyboards, in order to translate them into *technical specifications*.

It appears that the transition of information between the design phase and the development phase is a serious bottleneck (Boot & van Merriënboer, submitted). The intentions of an instructional design, described in the training blueprint and storyboards, are often not sufficiently represented in the technical specifications created by the producers. Time-consuming reviews and frequent discussions between instructional designers and software producers are often required to reach correct technical specifications, which are fully in line with the blueprint and storyboard. This sub-optimal translation process is deteriorated by the fact that many software experts are not necessarily experienced in specifying and creating *instructional* software programs. In absence of those reviews and discussions, the production process often results in an

unsatisfactory outcome, that is, flawed instructional software that requires correction afterwards ("design by debugging").

Recent software engineering methods attempt to overcome the translation problem by means of agile development methods. Agile methods emphasize *iteration* in the development process (see http://www.agilealliance.com). For instance, eXtreme Programming (XP) prescribes (a) revisiting preceding phases if information is insufficient, and (b) rapid prototyping of small but representative intermediate products for testing during development (Verstegen, 2003). However, such approaches are often not feasible due to the juridical and financial restrictions of outsourcing the production phase—increasingly applied in large (instructional) software projects—which separates the design and production phases in space and time. Such outsourcing to external parties, particularly if these parties are in foreign countries ("offshore outsourcing"), also limits information exchange due to language problems and cultural differences.

As a result, the development process relies heavily on the communicative quality of design documents such as training blueprints. An important question is therefore how such documents can be improved by instructional designers, to increase the probability of an optimal transfer of information from designer to producer. In the remaining parts of this Introduction, three fundamental variables of creating design documents are presented: Organization, detail, and standardization. Subsequently, the 3D-model for organizing design documents, based upon the three variables, is introduced. Then, an empirical study is presented comparing the effects of conventional design documents and improved design documents on the efficiency of the translation process and producers' satisfaction. The results of this study and, finally, their implications for future research and the practical field of instructional design are discussed.

Design documents

In the field of instructional software development, designers and producers lack a common, explicit notation system (Gibbons, Nelson, & Richards, 2000; Waters & Gibbons, 2004). A notation system is an embedded element of a design language and captures abstract ideas to create transferable designs (Gibbons & Brewer, 2005). Part of the reason why designers and producers use different languages and notation systems, even though they are discussing the same instructional software, is simply that they are interested in different aspects of the product and thus need to describe different features and functionalities (Nelson, 2003). The designer is mainly concerned with the content and the instructional strategies realized by the product, while the producer is mainly interested in its architecture and necessary data structures. Therefore, designers typically work with training blueprints and storyboards. A training blueprint consists of intermediate instructional design products, such as learning hierarchies, task classes, learning tasks, structures for scaffolding, cognitive feedback messages, and so forth. A storyboard consists of sketches of the interface lay-out and information for media selection and navigation.

Producers, in contrast, typically work with technical specifications consisting of highly detailed structural and procedural descriptions of the instructional software, such as program architectures, interaction patterns, navigation models, data types, information flows, reusable learning objects with metadata, and so on. For both designers and producers, the different sets of terms refer to different aspects of the same final product. But when training blueprints and storyboards are translated into technical specifications, the communication of ideas and the quality of the final product suffers if there is a mismatch of languages and notation systems (Nelson, 2003).

Three basic variables directly affect the quality of the translation from instructional design documents (blueprints and storyboards) to technical specifications, namely, the (1) organization, (2) level of detail, and (3) standardization of the design information. With regard to the *organization* of design information, the descriptions of different instructional and technical structures are often not meaningfully interrelated in conventional design documents. Adapting such documents to reflect changes in the design can be very difficult and laborious for the designers. Also, if producers face changes after the design phase, it will be very difficult for them to determine the effects of such changes for the technical specifications and the final product. For instance, the training of a problem-solving task can change because a new device so strongly supports the original problem-solving task that it becomes a routine task. This implies considerable implications for instruction (e.g., more emphasis on repetition of similar practice items combined with just-in-time information) as well as technical issues (e.g., different information to be displayed, different interactions, different feedback mechanisms, and so on).

The second variable is the level of *detail* of the design information. The level of detail in conventional design documents varies depending on the capabilities of the designer. For example, more capable designers will typically add more detail to instructional issues but not to technical issues. However, the level of detail should also depend on the needs of the receiver of the information, that is, the producer. For instance, to communicate between designers the application of delayed cognitive feedback following a particular learning task, a rather conceptual description will suffice. The designers will readily understand each other. But for a producer, much more detailed descriptions of timing, content, and presentation of feedback are needed to be able to specify and implement it as intended by the designer.

The third variable is the *standardization* of design information. In conventional design documents, designers express an instructional design mostly by textual expressions, supplemented with tables, lists, flowcharts, and graphics – all in a non-standardized manner. For producers, this leads to (a) semantic problems, as they may not fully understand the intentions of the designer, (b) interpretation problems, as they are left with too many degrees of freedom in creating the technical specifications, and (c) compatibility problems, as they cannot directly and (semi) automatically translate a design description into technical specifications.

We assume that the three basic variables (a) organization, (b) level of detail, and (c) standardization of design documents provide a starting point to improve the quality of the communication between designers and producers. Eventually, this will improve the quality of instructional software products.

The 3D-model

The 3D-model is established to support improving design documents. It consists of three dimensions, namely (a) *stratification*, (b) *elaboration,* and (c) *formalization,* based upon the variables discussed in the previous section. The three D's in the name reflect the three dimensions and are also an acronym for Developing Design Documents. Independent designers, or teams with designers and producers, may use the 3D-model to (a) analyze their design situation (e.g., what kind of designers and producers are involved? For what kind of training is the design made? Which support tools are available?) to determine the most optimal configuration of the 3D-model, and subsequently (b) use this configuration to stratify, elaborate, and formalize their design documents. Figure 3.1 presents the 3D-model in its full configuration, in which all dimensions are completely utilized.

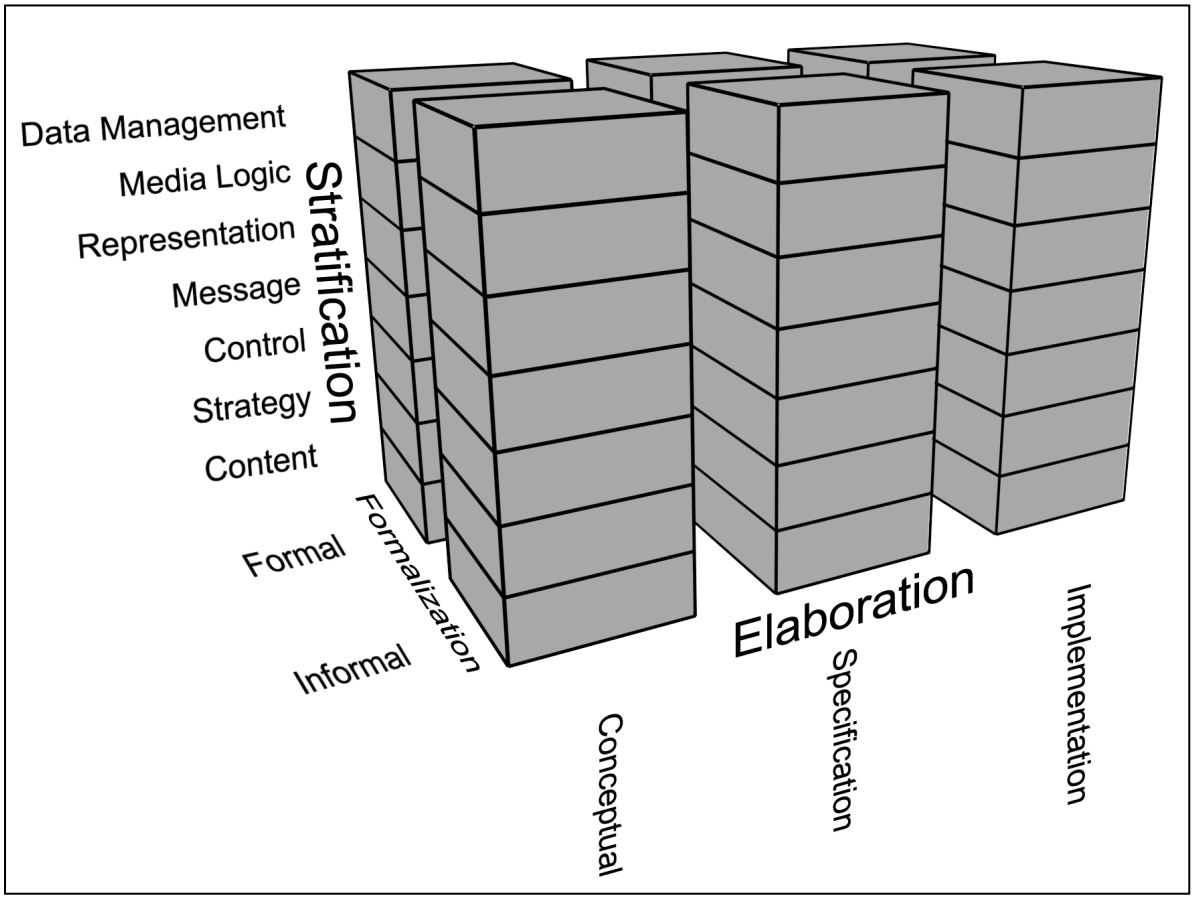

Figure 3.1. *The 3D-model for Developing Design Documents in its full configuration.*

First, in order to improve the organization of design documents, designers may stratify their designs in terms of a layered design architecture. For example, according to Gibbons' model of Design Layers (Gibbons, 2003), each complete instructional design is organized on seven, interrelated layers: Content, strategy, control, message, representation, media logic, and data management. Each layer is typified by the designer's selection of design languages pertaining to the solution of different instructional design sub problems. Together, the functional designs at the different layers, expressed in one or more design languages, make up the total design. The design of each layer may require different design activities, support tools, and specialists (see Table 3.1). For both designers and producers, stratification helps to identify the relations between the functionally-different instructional and technical structures, while at the same time staying cognizant of the need for integration within the complete design. Designers should therefore decide to which extend they are able to complete the stratification dimension, given their specific design situation.

Table 3.1
Objectives and Examples of the Seven Design Layers (Adapted from Gibbons, 2003)

Layer	Objective	Examples of activities	Examples of outcomes
Content	Define the content and structure of the domain ("what should be learned")	Task analysis, Content analysis, Concept mapping, Model analysis	Task hierarchies, Mental model descriptions
Strategy	Define the instructional design ("how should be learned")	Identification of whole-task practice and part-task practice, Definition of and sequencing of learning tasks, Definition of social relationships during instruction, Definition and sequence of time-event structures, Definition of roles, goals, and initiative-sharing during instruction	Task classes, Case descriptions, Feedback mechanisms
Control	Define the	Identification of	Content controls,

	command language given the learner for communication of actions and responses to the instructional source ("how can the user interact")	user actions, Definition of control space, Flow planning	Strategy controls, Administrative controls
Message	Define the message design ("what should be sensed")	Definitions of message structure, Composition of elements and rules	Message standards design for content
Representation	Define the representation design ("how should it be shown)	Media selection, Selection of production tools and methods	Layout standards, Media channel assignment, Media synchronization methods
Media-Logic	Define the software architecture ("how should the program be structured")	Definition of logic structure, Algorithms Creation, Learning objects definition	Modularity plan, Packaging method, Software platform selection, Maintenance plan
Data Management	Define the data management ("how should information, captured during instruction, be organized, analyzed, stored, and reported")	Defining administration processes, Data base selection, Definition of data items, capture, filtering, storage, analysis, interpretation, compilation, and sharing	Security plan, Billing methods, Metadata assignment

Second, in order to add sufficient detail to each layer, designers may elaborate their designs according to three different perspectives (Fowler, 2003). First, in a *conceptual* perspective, designers can describe the design more or less superficial and descriptive, reflecting the general direction of the design. Second, in a *specification* perspective, designers can describe the design more or less comprehensive and detailed, reflecting all design decisions. Third, in an *implementation* perspective, designers can describe the design more or less technical and meticulous. For both designers and producers, elaboration helps to determine the required minimum level of detail, depending on the

capabilities of the designer and the needs of the producer. Designers should therefore decide for each design layer to which extend they are able to progress along the elaboration dimension, given their design situation.

Third, in order to add sufficient standardization to the descriptions of each layer-perspective combination, designers may formalize their design descriptions by making their informal or formal design languages explicit. Formalization helps to add rigor to a design to promote unequivocal understanding of both designers and producers. Designers should strive for (combinations of) formal languages, but depending on the capabilities of the designer and the needs of the producer, they can also select (combinations of) informal languages. Such a language can be specific for a particular layer, for instance, informal languages such as event-and-control flow diagrams for the control layer, and wire frames of layouts for the representation layer. Or it can be specific for a particular perspective, for instance, an informal language such as plain text for the conceptual perspective; the Unified Modeling Language (UML; Fowler, 2003) for the specification perspective, and the Extended Markup Language (see www.w3.org/XML) for the implementation perspective. Designers should therefore decide for each design layer and each perspective which (in)formal design languages are suitable, given their design situation.

The application of the 3D-model will result in a specific configuration for each different design situation. This is expected to result in a more efficient translation process and a higher producers' satisfaction with the design documents than with conventional design documents. The current study is conducted to verify this claim. It is hypothesized that compared to conventional design documents the improved documents lead to a better understanding by the producers and require less time and perceived cognitive load to reach this understanding.

Method

Participants

Sixteen students from Utah State University's Computer Science department participated in this study, acting as producers of instructional software. They were randomly assigned to either the conventional documents group (n = 8) or the improved documents group (n = 8). All participants received a compensation of $ 20.

Materials

Design documents. The conventional and improved design documents were on an identical topic, learning to drive a car, and had an identical function, providing input for the technical specification process for an advanced car-driving educational simulation (see van Emmerik, 2004). With respect to ecological validity, reviews of expert instructional designers and producers, not related to the study and blinded for condition, indicated that the design

documents were representative for documents used in professional training organizations.

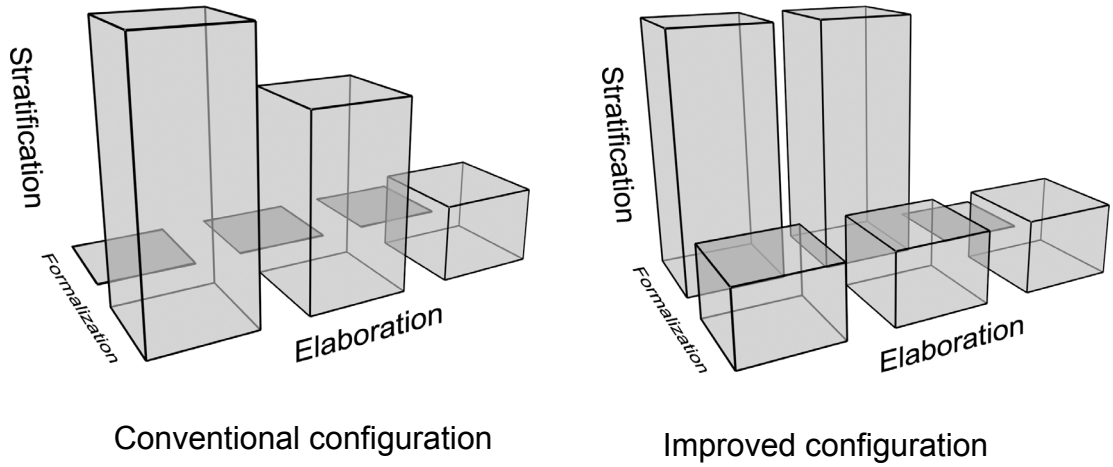

Figure 3.2. *Configurations of the 3D-model for the conventional and improved design documents.*

Figure 3.2 describes the difference between the conventional and improved design documents in terms of configurations of the 3D-model. In the conventional design document, the value on the "formalization" dimension is always *informal*, thus no formal representations are used. The value on the "elaboration" dimension is *implementation* for the content and strategy layers; *specification* for the control, message, and representation layers; and *conceptual* for the media logic and data management layers. This configuration reflects the traditional approach towards design documents. The content and strategy layers are described as a training blueprint specified in the four-component instructional design model (van Merriënboer, 1997; van Merriënboer, Clark, & de Croock, 2002), containing typical instructional design information such as a task-hierarchy, a list of learning objectives, descriptions of learning tasks, and an overview of the whole training program. The control, representation, media logic, and data management layers are described with storyboards, containing typical instructional software information on user-interfaces, navigation, interaction, and program-flow. The information was described by text, tables, flowcharts, and drawings according to best practices and guidelines from the literature (e.g., Driscoll, 1998; Kruse & Keil, 2000).

In the improved design document, the values on the "formalization" dimension are both *formal* and *informal*, thus both kinds of representations are used. For the informal representations, the values on the "elaboration" dimension are *conceptual*, *specification*, and the values on the content and strategy layers are *implementation*. For the formal representations, the values on the

"elaboration" dimension are *specification* and *conceptual* for the layers content up to data management. This configuration reflects the use of the 3D-model to stimulate and support designers to stratify, elaborate, and formalize design documents more than they usually do. As in the conventional design document, the informal representations were described by a training blueprint according to the four-component instructional design model and storyboards. The formal representations were described by UML diagrams.

Measurements

Background questionnaire. The background questionnaire was used to collect information about the participants' (a) experience with car driving, related to the topic of the design documents (ownership of drivers license, years of driving experience); (b) level of education; (c) number of familiar object-oriented programming languages, and (d) familiarity with Object-Oriented Programming (OOP), Object Oriented Modeling (OOM), and UML.

Specification questionnaire. The ability to translate the design document into technical specifications, defined as the *results* of the translation process, was measured by the specification questionnaire. It consisted of 25 open questions, each question on one printed page with sufficient space to note down the answer. There was no time limit for answering the questions. Each question addressed a particular aspect of translating the design document into technical specifications. For instance, the participants had to distill from the design document how many databases should be used in the instructional software; what the consequences would be from changing text-based messages into audio-based messages (the so-called "ripple effect"); how a particular program flow should be implemented; what it meant if just-in-time information would be applied in a particular learning task; where the producer would need a subject matter expert to provide additional domain information; which instructional design components should be implemented as reusable learning objects, and so forth. Based on a checklist with correct answers, two reviewers rated all items as correct or incorrect (the Intra Correlation Coefficient, ICC, is .94, which is good, Fleiss, 1981).

Cognitive load questionnaire. This questionnaire measured the perceived cognitive load for each question in the specification questionnaire, defined as part of the *costs* of the translation process. It used the standard 9-point rating scale developed by Paas (1992; see also Paas, Tuovinen, Tabbers, & van Gerven, 2003). The rating scale was included at the bottom of each page of the specification questionnaire, and ranged from 1 = "very, very low perceived load" to 9 = "very, very high perceived load". The ICC of the questionnaire is .89, which is good.

Satisfaction questionnaire. This questionnaire measured the participants' satisfaction with the design documents. It contained six statements that had to be rated on a 9-point scale (ranging from 1 = "very, very low" to 9 = "very, very high"). The statements concerned (a) the effort that needs to be invested in the technical specification process, (b) the capability to create technical

specifications, (c) the perceived completeness, (d) the level of detail, (d) the understandability, and (e) the quality of the design document.

Procedure

First, the participants were asked to fill out the background questionnaire. Subsequently, they were asked to study either the conventional or the improved design document for exactly 50 minutes. During this period, they were allowed to make notes for later use. Then, they had to fill out the specification questionnaire. The 25 pages of the questionnaire were filled out one by one, allowing the experimenter to note down the time on task for each question (in units of half minutes). In addition to perceived cognitive load, time was defined as another part of the *costs* of the translation process. After finishing each question, the participants filled out the cognitive-load questionnaire at the bottom of the page and gave it to the experimenter. Immediately after answering the final question the participants filled out the satisfaction questionnaire.

Data Analysis

T-tests for independent samples are used to test for differences between the conventional and improved document groups. The relative efficiency of both groups is calculated using the 3D-efficiency approach of Tuovinen and Paas (2004). In this approach, efficiency is defined as the difference between standardized *results* (in this study the quality of technical specifications (QTS) as a result of the translation from design documents) and standardized scores for perceived cognitive load (PCL) and time on task (TT), reflecting the *costs*. In a three dimensional Cartesian space, efficiency is the perpendicular distance between a point in that space and a plane that represents an efficiency of zero, and determined by the equation:

$$E = \frac{QTS - PCL - TT}{\sqrt{3}}$$

Results

In the conventional documents group, participants' education was computer science on the Bachelors level (3 out of 8), Masters level (4 out of 8), or PhD level (1 out of 8). All participants had a driver license, and their mean car driving experience was 7.88 years (SD = 6.01). In the improved documents group, participants' education was computer science on the Bachelors level (6 out of 8) or Masters level (2 out of 8); 6 of the 8 participants had a driver license, and their mean car driving experience was 6.75 years (SD = 5.25). As can be seen in Table 3.2, participants have experience in at least two object-oriented programming languages and rated their experience with OOP, OOM, and UML above average. There were no significant differences between groups.

Table 3.2
Means and Standard Deviations of Proficiency with Programming Languages and Ratings on Experience with Object-oriented Software Development

	Conventional design documents group (n = 8)		Improved design documents group (n = 8)	
	M	*SD*	*M*	*SD*
1. # of familiar OOP languages	2.37	0.92	1.87	0.99
2. Object-Oriented Programming [a]	7.13	0.99	6.63	1.99
3. Object-Oriented Modeling	6.75	0.88	6.37	1.41
4. Unified Modeling Language	5.75	2.52	5.25	2.31

[a] Questions 2-4 are rated on a 9-point scale (1 = "very, very low"; 9 = "very, very high").

Table 3.3 presents the main results on the quality of the technical specifications (i.e., agreement with the design document), time on task, perceived cognitive load, and relative efficiency of the translation process. The quality of the technical specifications is higher in the improved documents group (M = 17.18 on a scale with a maximum of 25, SD = 1.94) than in the conventional documents group (M = 12.25, SD = 2.35; t = 4.58, p < .001). The mean time on task per question is lower in the improved documents group (M = 2.75 minutes, SD = .71) than in the conventional documents group (M = 3.46 minutes, SD = 0.89; t = 1.77, p < .05). The perceived cognitive load does not differ between groups (t = 0.88, p > .39); the improved documents group scored a mean of 4.06 on a 9-point scale (SD = 1.10) and the conventional documents group scored a mean of 4.43 (SD = .42). As expected, the efficiency of the translation process is significantly higher in the improved documents group (M = .28, SD = .37) than in the conventional documents group (M = -.28, SD = .38; t = 3.03, p < .01).

Table 3.3
Means and Standard Deviations for Measures of the Transition Process

	Conventional design documents group (n = 8)		Improved design documents group (n = 8)	
	M	*SD*	*M*	*SD*
1. Quality of production (0 – 25)	12.25	2.35	17.18	1.94
2. Mean time per question (mins.)	3.46	0.89	2.75	0.71
3. Mean perceived cognitive load per question [a]	4.43	0.42	4.06	1.10
4. Efficiency of specification process	-.28	.38	.28	.37

[a] Item 3 is rated on a 9-point scale (1 = "very, very low"; 9 = "very, very high").

Table 3.4 presents the results on participants' satisfaction with the design documents. In general, the participants were reasonably satisfied. There are no significant differences between the groups.

Table 3.4
Means and Standard Deviations for Satisfaction with Design Documents

	Conventional design documents group (n = 8)		Improved design documents group (n = 8)	
	M	*SD*	*M*	*SD*
1. What is your invested effort? [a]	5.25	1.04	5.87	1.45
2. How is your ability to create technical specifications based upon the design documents?	5.87	1.25	5.25	1.83
3. How is the level of completeness of the design documents?	4.87	1.81	6.00	0.75
4. How is the level of detail of the design documents?	5.25	1.48	6.37	1.59
5. How is your understanding of the design documents?	6.13	1.25	6.50	0.93
6. How is the quality of design documents?	5.63	1.99	5.87	1.23

[a] Items are rated on a 9-point scale (1 = "very, very low"; 9 = "very, very high").

Discussion

This study investigated means to improve the efficiency of the translation process between the design phase and the production phase. The results show that the application of a structured, three-dimensional approach by designers, helps producers to specify technical specifications that are more in agreement with instructional design documents (i.e., training blueprint and storyboard). By improving these documents, less of the designers' intentions are "lost in translation," preventing the specification and production of sub-optimal products. Developing design documents while supported by the 3D-model results in a higher efficiency of the translation process, reflecting better results in combination with less time and perceived cognitive load.

This study assumed that primarily instructional designers are in a position to enhance the efficiency of the translation process through the improvement of instructional design documents, because producers cannot correctly judge the quality of those documents. The results of this study are in line with this assumption, because no relation between producers' satisfaction and the nature of the design documents was found. The lack of differences in satisfaction with conventional and improved design documents might be caused by the fact that the producers in this study were not professional instructional software developers but students. However, the participants indicated to be experienced with several programming languages and techniques. Most instructional software development projects will use producers with equal or even less experience than the participants in our study.

Developing design documents according to the 3D-model suggest three lines for future research. The first line pertains to variations on the current study. For instance, the effects of different configurations of the 3D-model (compare Figure 3.2) on the efficiency of the translation process may be studied, as the optimal configuration is likely to be dependent on the specific design situation. As another example, the creation of the design documents may be varied by changing the characteristics of the instructional designers in terms of experience, educational sector (e.g., formal schooling, military, government, industry), subject matter domain (social, technical), and so forth.

The second line pertains to the role support tools can play for the configuration of the 3D-model. For instance, new tools such as ADAPT-IT (De Croock, Paas, Schlanbusch, & van Merriënboer, 2002, also see www.enovateas.com) support the easy creation of design documents in a structured manner. With respect to the dimensions of the 3D-model, ADAPT-IT helps designers to create design documents that are both formal and informal, are elaborated at the conceptual and specification level, and describe the content and strategy layers. Future research may either investigate the contribution of support tools to the creation of design documents, or use the 3D-model to develop new tools that take all three dimensions into account.

The third research line pertains to the role standardized instructional design languages may play for the instantiation of the 3D-model. For instance,

new languages such as IMS Learning Design (IMS-LD; Koper & Tattersall, 2005) and E2ML (Botturi, in press) offer opportunities to specify particular (combinations of) cells in the 3D-model. This is particularly relevant for formal representations at the elaboration levels 'specification' and 'implementation.' Further research should indicate to which extend these new languages may contribute to the quality of the design documents and the efficiency of the translation process.

An important practical implication of this study concerns the schooling of instructional designers. The results of this study imply that producers are not in a good position to improve design documents, because they have difficulties in judging the quality of these documents. In addition, they cannot always ask the designer for clarification (e.g., in the case of –offshore—outsourcing). This puts the responsibility for improving design documents predominantly on designers. Besides being knowledgeable and skilled in traditional instructional design activities such as domain and task analysis, strategy selection, and media selection (see Richey, Fields, & Foxon, 2001), our results indicate that instructional designers need to become proficient in at least three new activities. First, they should be able to stratify instructional design documents to describe aspects associated with design as well as production. Second, they should be able to decide for each layer how much detail is required for unequivocal understanding of the design by producers. Finally, they should be able to represent their designs in formal design languages such as UML, IMS LD, or E2ML. Support tools may help them to perform their new activities. Due to limitations in budget and time, formal education will not always be feasible. Communities of practice might offer an alternative option, because they provide designers and producers with a platform to discuss each others information and training needs.

References

Boot, E. W., & van Merriënboer, J. J. G. (submitted). Improving the development of instructional software: Three building-block solutions to interrelate design and production.

Botturi, L. (in press). E2ML: A visual language for the design of instruction. *Educational Technology, Research and Development.*

De Croock, M. B. M., Paas, F., Schlanbusch, H., & van Merriënboer, J. J. G. (2002). ADAPT-IT: ID tools for training design and evaluation. *Educational Technology, Research and Development, 50*(4), 45-58.

Dick, W., & Carey, L. (1996). *The systematic design of instruction.* New York: HarperCollins.

Driscoll, J. (1998). *Web-based training: Tactics and techniques for designing adult learning*. San Francisco, CA: Jossey-Bass/Pfeiffer.

Fowler, M. (2003). *UML distilled: A brief guide to the standard object modeling language.* Boston, MA: Addison-Wesley Professional.

Fleiss, J. L. (1981) *Statistical methods for rates and proportions (2nd Ed.).* New York: Wiley.

Gibbons, A. S. (2003). What and how do designers design? A theory of design structure. *Tech Trends, 47*(5), 22-27.

Gibbons, A. S., & Brewer, E. K. (2005). Elementary principles of design languages and design notation systems for instructional design. In J. M. Spector, C. Ohrazda, A. Van Schaack & D. Wiley (Eds.), *Innovations to instructional technology: Essays in honor of M. David Merrill* (pp. 111-129). Mahwah, NJ: Lawrence Erlbaum Associates.

Gibbons, A. S., Nelson, J., & Richards, R. (2000). The nature and origin of instructional objects. In D. A. Wiley (Ed.), *The instructional use of learning objects* (pp. 25-58). Bloomington, IN: Association for Educational Communications and Technology.

Jochems, W., van Merriënboer, J. J. G, & Koper, R. (Eds.). (2004). *Integrated e-learning: Implications for pedagogy, technology, and organization.* London, UK: RoutledgeFalmer.

Koper, R., & Tattersall, C. (Eds.). (2005). *Learning design: A handbook on modeling and delivering networked education and training.* Berlin, Germany:Springer-Verlag.

Kruse, K., & Keil, J. (2000). *Technology based learning: The art and science of design, development, and delivery.* San Francisco, CA: Jossey-Bass Pfeiffer.

Merrill, M. D. (2002). First principles of instruction. *Educational Technology, Research and Development*, *50*(3), 43-59.

Nelson, J. S. (October, 2003). Separating the media logic layer: An argument for a layered theory of design. *Roundtable session at the annual conference of the Association of Educational Communications and Technology (AECT).* Anaheim, CA.

Paas, F. (1992). Training strategies for attaining transfer of problem-solving skill in statistics: A cognitive-load approach. *Journal of Educational Psychology*, *84*, 429-434.

Paas, F., Tuovinen, J., Tabbers, H., & van Gerven, P. W. M. (2003). Cognitive load measurement as a means to advance cognitive load theory. *Educational Psychologist, 38,* 63-71.

Richey, R. C., Fields, D. C., & Foxon, M. (Eds.) (2001). *Instructional design competencies: The standards (3rd Ed.).* Syracuse, NY: ERIC Clearinghouse on Information and Technology and the International Board of Standards for Training, Performance and Instruction.

Salden, R. (2005). *Dynamic task selection in aviation training.* Unpublished PhD Dissertation, Open University of the Netherlands, Heerlen.

Tabbers, H. K. (2002). *The modality of text in multimedia instructions: Refining the design guidelines.* Unpublished PhD Dissertation, Open University of the Netherlands, Heerlen, The Netherlands.

Tuovinen, J. E., & Paas, F. (2004). Exploring multidimensional approaches to the efficiency of instructional conditions. *Instructional Science, 32,* 133-152.

Van Emmerik, M. L. (2004). *Beyond the simulator: Instruction for high-performance tasks.* Unpublished PhD Dissertation, University of Twente, Enschede.

Van Merriënboer, J. J. G. (1997). *Training complex cognitive skills: A four-component instructional design model for technical training.* Englewood Cliffs, NJ: Educational Technology Publications.

Van Merriënboer, J. J. G., & Sweller, J. (2005). Cognitive load theory and complex learning: Recent developments and future directions. *Educational Psychology Review, 17*(2), 147-177.

Van Merriënboer, J. J. G., Clark, R. E., & De Croock, M. B. M. (2002). Blueprints for complex learning: The 4C/ID-model. *Educational Technology, Research and Development, 50*(2), 39-64.

Verstegen, D. M. L. (2003). *Iteration in instructional design: An empirical study on the specification of training simulators.* Unpublished PhD Dissertation, Utrecht University, Utrecht.

Waters, S. H., & Gibbons, A. S. (2004). Design languages, notation systems, and instructional technology: A case study. *Educational Technology, Research and Development, 52*(2), 57-68.

Chapter 4

Novice and Experienced Instructional Software Developers: Effects on Materials Created with Instructional Software Templates[3]

Abstract

The development of instructional software is a complex process, posing high demands to the technical and didactical expertise of developers. Domain specialists rather than professional developers are often responsible for it, but authoring tools with pre-structured templates claim to compensate for this limited experience. This study compares instructional software products made by developers with low production experience (n = 6) and high production experience (n = 8), working with a template-based authoring tool. It is hypothesized that those with high production experience will be more productive and create software with a higher didactical quality than those with low production experience, whereas no differences with regard to technical and authoring quality are expected. The results show that the didactical quality was unsatisfactory and did not differ between groups. Nevertheless the templates compensated for differences in experience because the technical and authoring quality was equal for both groups, indicating that templates enable domain specialists to participate successfully in the production process.

[3] This chapter will be published as: Boot, E., & van Merriënboer, J. J. G. (in press). Novice and experienced instructional software developers: Effects on materials created with instructional software templates. *Educational Technology, Research and Development*.

The development of instructional software is a costly and time-consuming process (Tennyson & Barron, 1995). This development bottleneck, concerning the instructional design and software production, is becoming more and more serious because current trends in e-learning, such as just-in-time and just-enough learning (Rosenberg, 2000), increase the need for large amounts and many different kinds of instructional software, to be created for very specific needs, in short periods of time. The technical automation of development is only beneficial when it concerns a large volume of instructional software for the same topic (Spector & Muraida, 1997), such as software for teaching (foreign) languages or information technology skills. But as most knowledge domains are more specific, instructional software needs to be developed custom-made. This is typically done by domain specialists, because they already possess the necessary domain knowledge and have easy access to relevant—multimedia—resources (Spector & Muraida, 1997), and professional instructional designers and software producers are not easily available or too expensive to hire. Such domain specialists are proficient in one or more task domains. In organizations such as the military, typically senior domain specialists are also tasked with teaching about their domain, and designing traditional instructional materials such as readers, syllabi, course plans, workbooks and so forth.

The didactical quality of instructional software (defined as the extent to which desired learning outcomes are attained in an efficient manner) is of utmost importance because technical quality (defined as the extent to which the software takes care of the input, information processing, and output as intended) alone is necessary but not sufficient to stimulate the desired learning processes. Two approaches can be used to assess the didactical quality. First, an empirical approach which can determine if learners who use the software indeed reach the learning objectives specified beforehand. Secondly, an analytical approach that can determine to which degree particular instructional principles are embedded in the software. For example, Merrill (2002) describes five 'first principles of learning' (2002): The extension to which these principles are implemented determines the didactical quality of the software. The principles refer to (1) the use of real-life *problems* as the driving force for learning; (2) the proper *activation* of relevant prior knowledge; (3) the *demonstration* of useful problem-solving approaches and procedures; (4) the practical *application* of those approaches and procedures by the learner, and (5) the *integration* of what has been learned into real-world activities.

However, principles and guidelines from instructional design theories are often not very concrete and difficult to apply unequivocally in developing instructional software. As Goodyear (1997) states: "There are many gaps between what the prescriptive literature on instructional design would have us believe and the vicissitudes of design and production practice" (p. 83). For example, it is mainly the way that the five principles mentioned above are applied rather than their mere presence that determines the didactical quality of software. Consequently, more powerful ways to support domain specialists with their development of instructional software are needed. Our main research

question is if so-called *instructional software templates* can adequately support domain specialists who develop (i.e., design and produce) instructional software. In this Introduction, first, different approaches to support domain specialists are presented, second, the use of templates in modern development tools is described, and third, the hypothesized effects of such templates are discussed. Then, we present an empirical study in which the quality of instructional software developed by novice and experienced domain specialists is compared.

Supporting Domain Specialists

How can the instructional software development process by domain specialists be made more efficient (i.e., faster, cheaper, easier) and more effective (i.e., improved didactical quality of the final product)? One way is to provide formal training to improve both design and production skills for creating instructional software. However, such training is often not possible due to a lack of budget, time, or suitable training programs. Another way is to provide so-called 'authoring tools' (Locatis & Al-Nuaim, 1999), which intend to make the development process from paper-based instructional blueprint to concrete instructional software faster and easier, while still creating effective and appealing instruction (Merrill, 1997). For instance, such tools hide complex programming code by providing intuitive interfaces and pre-structured program components. Familiar tools are Macromedia's Authorware and Asymetrix' Toolbook, but current Learning Content Management Systems (LCMSs) also begin to provide build-in authoring facilities (Chapman, 2003). Vendors often claim that their tools allow for the easy and rapid creation of instructional software with a high didactical quality – even when developers have little or no experience. Thus, authoring tools seem to provide a good solution for supporting domain specialists who need to develop instructional software but have no or little experience with that.

An inventory study for the Royal Netherlands Army analyzed the process of instructional software development by novice developers such as domain specialists who used different authoring tools (Boot & van Rooij, 1999). The domain specialists were mostly former instructors with considerable experience in teaching and developing traditional instructional materials, but with limited multimedia and instructional software development experience. The study showed that the development of instructional software was much harder for the domain specialists than expected, mainly because of (1) authoring problems, (2) technical problems, and (3) didactical problems.
First, it became clear that despite the use of authoring tools the production process still required considerable programming knowledge, which the domain specialists seldom possessed (see also Merrill, 1997). Offering a collection of pre-structured pieces of programming code and ready-made user interfaces was not enough. The authoring process still required insight in how to structure a program, how to deal with data variables, how to implement a navigation structure, how to apply an effective interaction design, and so forth. As a

consequence, the resulting products showed a lack of standardization in the uniformity of program structures and user-interfaces. In addition, most projects did not meet their deadlines and the quality of the final products greatly varied.

Second, an interesting finding was that the domain specialists seldom reused existing multimedia resources in order to make the production process more cost-efficient. Reuse was impeded by the lack of technical knowledge of file formats, sizes, resolutions, and interfacing, required to (re-)combine multimedia resources and programming structures. This problem is further complicated by the fact that Internet technology, operating systems, and authoring tools are constantly being changed and updated, and are also becoming increasingly more advanced and complex. The domain specialists also lacked adequate tools to support them in reusing resources efficiently.

Finally, it was found that even domain specialists with high instructional design experience, had great difficulties to apply didactical models that made an optimal use of the new possibilities offered by multimedia systems and instructional software. Possibly, traditional didactical models differ too much from new models for 'digital learning and instruction' to allow them to make the necessary transition (Simons, 2002; van Merriënboer, 2003). More specifically, it is difficult for domain specialists to implement multimedia instruction for at least three reasons. First, to embed didactical models in instructional software, the design must be quite detailed. For example, feedback on learner actions has to be anticipated beforehand because the developer cannot intervene in the learning process 'on-the-fly' as a teacher can do in a classroom. Second, domain specialists may not be aware of the didactical models that are (implicitly) supported by authoring systems. For example, in a 'drill and practice' or 'learning by doing' setting, simulated environments can offer authentic opportunities for an extensive practice of skills. Or, in a 'mastery learning' setting, decisions with regard to mastery and subsequent measures can be based on a process of continuously, non-intrusive tracking and tracing of learning results. Third, domain specialists must be extremely careful with implementing combinations of different visual and auditory representations. For example, narration may work well for explaining a system-paced streaming animation but be less effective than a visual explanation if the learner is able to set the pace of the animation (for more examples, see Mayer, 2001; Sweller, van Merriënboer, & Paas, 1998).

Template-based Development Tools

Commercial vendors currently provide so-called 'zero-programming' tools that are claimed to solve the authoring, technical, and didactical problems in both instructional design and software production. These tools are characterized by the use of *instructional software templates*. The concept of such templates can be traced back to earlier developments in the field of software engineering. As soon as programmers discovered that it was helpful to automate certain routine and repetitive task aspects they created template-like structures that contained parts of a program, which only had to be instantiated

with new data and adapted towards new uses. Subsequently, the software methodology of Object Oriented Programming (OOP) focused on improving programming by the decomposition of software into independent, reusable units that functioned like LEGO bricks (Ackermann, 1996; Booch, 1994). Another important feature of OOP is its use of design patterns (Gamma, Helm, Johnson, & Vlissides, 1995). Based on the assumption that in developing software one encounters many recurrent problem situations that require comparable solutions, these patterns provide standard solutions to common software development problems. Working in the tradition of this building-block approach, instructional software templates should potentially be able to provide support on the authoring, technical, and didactical level.

On the authoring level, instructional software templates offer pre-structured 'moulds' of instructional software. By analogy with OOP, the templates include different didactical objects. Such objects can be higher-level objects, including empty lesson structures, default navigation methods, and generic graphical user-interfaces, as well as lower-level objects, including structures to create practice items, test questions, examples, cases, feedback, and learner support. The screens that present the developer an easy-to-use interface to templates (i.e., *wizards*) guide the proper structuring and programming of instructional software.

On the technical level, instructional software templates automatically produce instructional software that is compliant with current technologies and operating systems. The vendor only needs to regularly update the templates in order to ensure compliance with the latest technical possibilities. Wizards can guide the proper embedding of multimedia objects into templates.

On the didactical level, instructional software templates provide didactical design patterns, either originating from instructional theory or from best practices. Default structures may, for instance, present drill-and-practice, concept learning, mastery learning, and case-based learning, thereby explicitly gearing design with production. Wizards can guide the proper application of these structures.

The Effects of Instructional Software Templates

The use of instructional software templates is not new in the field of instructional technology (e.g., see de Jong, Limbach, Gelleveij, Kuyper, Pieters, & van Joolingen, 1999; Cline & Merrill, 1995; Merrill & ID2 Research Group, 1998; Tennyson et al., 1995; van Merriënboer & Martens, 2002). But what is new is the appearance of a great number of commercial template-based tools and their widespread application in large development projects. As argued above, these tools may have the potential to overcome some or all of the problems associated with the development of instructional software, but until now very little is known about their actual benefits in real-life design projects.

In the template-based authoring tools, there is typically a strong emphasis on authoring and technical support (e.g., see WBTIC, 2004). Didactical support is largely neglected, and if it is mentioned at all, it is based on a highly

traditional information-transmission view on instruction resulting in linear and quite passive learning experiences. One reason for not supporting didactical models or only supporting overly simplified didactical models may be that it is much harder to give didactical support than to give authoring and technical support. Consequently, developers may experience limited didactical support and are biased or even encouraged to apply traditional didactical models whilst designing and producing instructional software. Thus, whereas in general tools cannot be blamed, the main impact of instructional software templates may be that they support the production of mediocre instructional software with a low didactical quality in a more efficient manner.

The present study examines if and how instructional software templates support domain specialists with low and high experience in producing instructional software. The domain specialists in these novice and expert groups are asked to perform the development task of producing an instructional software product by means of a typical set of templates. First, we hypothesize that, despite the support from the templates for the developers in the inexperienced group, the developers in the experienced group will produce *more* instructional software because they profit from their previous hands-on experiences with computers, development tools, and authoring systems. Second, we hypothesize that developers in both groups will produce final products with an equal *authoring and technical quality*. The templates are expected to support the experienced group while they conduct their regular authoring (e.g., entering the input, information processing and output facilities in the software) and technical (e.g., creation and embedding of multimedia files) development activities, and also to successfully scaffold the inexperienced group so that they can perform those activities up to the same level. Note that authoring and technical activities strongly interact, and are therefore combined. Third, we hypothesize that the developers in the experienced group will produce final products with a higher *didactical quality*, because they are more efficient in performing the authoring and technical task aspects and thus have more time to pay attention to (a) the instructional design, and (b) how to implement the design into the production process and product. Furthermore, it is expected that both groups feel they receive *pre-structuring support* from the instructional software templates, and this results in a faster development process and in final products with a sufficient didactical quality.

Method

Participants

Fourteen educational developers of the Royal Netherlands Army participated in the study on a voluntary basis. Their mean age was 47 years (*SD* = 4.20). All participants were experts in a particular military knowledge domain. They were developers with different experience in creating traditional instructional materials and/or instructional software. They expressed their teaching experience on a 5-point Likert scale as 2.92 (*SD* = 1.04; 1 = ‘not at all’

and 5 = 'very much'). The participants had little or no experience in using instructional software as a learner, or for instructional purposes as a teacher. They were highly motivated to participate in the study because this was one of the few opportunities during their career to learn about new, innovative ways of developing instructional software.

They were divided in a novice group (n = 6) and an experienced group (n = 8) based upon their experience with producing instructional software. This experience was measured by adding (a) number of years of experience in producing instructional software x (days per week developing instructional software / 5), (b) number of products developed, (c) number of years of experience with the authoring tool Authorware, and (d) number of years of experience with programming. This practical measurement yields a possible score between 0 and 46. Participants were regarded novice if they scored lower than 4, and experienced if they scored higher than 4. Experience was 0.42 for the novice group (SD = 0.80) and 15.90 for the experienced group (SD = 9.65), Mann-Whitney's $U(N = 14) = 0$, $p = .002$. The number of years of programming experience was checked too. The novice group had 0.58 years of programming experience (SD = 1.20) and the experienced group had 6.00 years of programming experience (SD = 3.07), $U(N = 14) = 1.50$, $p = .004$.

Materials

CBT Generator. The set of instructional software templates used in this study is part of an authoring tool called the CBT Generator (version 1.0). This support tool was specifically created for the Royal Netherlands Army to support military domain specialists in developing instructional software. It is based upon instructional software templates created by means of the Knowledge Objects in Macromedia Authorware© (version 5.2). The CBT Generator offers support with respect to authoring aspects, technical aspects, and didactical aspects – corresponding as closely as possible with the development procedures, best practices, and terminology used within the training sector of the Royal Netherlands Army.

First, authoring support is offered for structuring and programming the instructional software according to best practices in authoring tools. Wizards were offered to configure pre-structured pieces of 'empty' instructional software (i.e., templates), a standard graphical user-interface was provided, and possibilities for tracking and tracing of the learning process were given.

Second, technical support was offered with regard to embedding multimedia resources according to particular technical formats and presentation strategies. Wizards were offered to enter and configure—different combinations of—audio, text, and video in the templates.

Third, didactical support was embedded in the templates, to implement the instructional design according to guidelines used in the Royal Netherlands Army (e.g., Gagné's Nine Events, 1979; the guidelines presented by Leshin, Pollock, & Reigeluth, 1992). For example, guidelines pertained to the use of (a) teasers to gain attention; (b) different types of learning activities (12 types are available in the CBT Generator) to promote an active learning process; (c)

different sequencing models; (d) adaptive branching based upon a learner's progress; (e) different levels of learner control; (f) different kinds of feedback; (g) extra learner support through glossaries and help-functions, and (h) formative and summative tests for assessment of learning. The didactical elements mentioned above are available in four different categories in order to compose the actual learning tasks. The first category contains information elements to present the learner information (1 available type), the second category contains instructional elements to present the learner explanations (12 available types), the third category contains question elements to ask the learner to solve a problem or a case (12 available types), and the fourth category contains learner-evaluation elements to assess the learning process in a formative or summative fashion (8 available types). Note that the authoring, technical, and didactical aspects are implemented in a rather prescriptive manner in order to provide adequate pre-structuring support for developers to implement interactive and varied, flexible instruction. Wizards were offered to select and configure the instructional design implementations.

Tasks. All participants received the same paper-based assignment. First, they were given the task of developing (i.e., design and produce) an instructional software package for a training program in 'military first aid', using the CBT Generator. This topic was familiar to the participants because it is basic knowledge for all military personnel. Thus, possible initial differences in domain knowledge between experimental groups were prevented. Second, the assignment specified the time-constraints, the topic of the course, and the working procedure (i.e., time schedule, questionnaires that had to be filled out, etc.). Third, the participants were also told the criteria, in terms of quantity and authoring, technical and didactical quality, which will be used for the reviews on product quality. Fourth, for the instructional design, the target group characteristics and the learning objectives were provided. Fifth, for the software production, a list was provided with all available multimedia resources (pictures, texts, audio files, etc.) that could possibly be useful for developing the course. The participants were explicitly encouraged to reuse multimedia resources that typically are available in their normal work situation, such as fragments of manuals, pieces of information from the Internet, promotional materials from the Royal Netherlands Army, available CD-ROM materials, and so forth. Next to the assignment, a fully worked-out example was provided of a representative instructional software package about the topic 'introduction of the Euro', created by developers who were experts in using the CBT Generator. This worked-out example showed all possible development features of the available instructional software templates.

Development environment. The participants worked in separate rooms, equipped with a stand-alone multimedia computer with Microsoft Windows© and the CBT Generator. The multimedia resources were provided on a separate CD-ROM. Multimedia editing programs were pre-installed on the computer to enable the preview of multimedia resources. With these tools it was also possible to convert the format or the resolution of the resources. However, as the

purpose of the study was not to test if the participants were able to edit the multimedia materials, most resources could be used in the CBT Generator without modification.

Measurements

Background questionnaire. The background questionnaire was used to collect information on (1) the participants' experience with programming instructional as well as non-instructional software (both in years of experience, daily experience, number of programmed applications, and usage of authoring tools); (2) their age; (3) their job profile (e.g., developer of traditional educational materials or instructional software); (4) their prior education (years and level of general education, particular Army courses they participated in), and (5) their experience with instructional software and educational development projects (in years and number of projects). The two experimental groups differed with regard to—instructional—programming experience, but were expected to be similar with regard to the other background variables.

Product reviews. Three reviews on the instructional software products took place, pertaining to (1) the quantity of materials produced, (2) the quality of the authoring and technical aspects of the materials, and (3) the quality of the didactical aspects of the materials.

First, the quantity was determined by the experimenter, as time on development task and length of the final product as the number of pages, and as the average number of edited didactical elements (informational, instructional, question, and learner-evaluation) on each page. Pages and didactical elements that were created but not filled out, or otherwise not used, were left out from the counting process.

Second, the quality of the authoring and technical aspects of the materials were combined and measured in two ways. In the first measurement, the proper functioning of the product was determined by the experimenters by running the software and examining its behavior. The behavior was rated as either insufficient (problems with running the software) or sufficient (no problems). Extra care was taken to ensure that reported problems were caused by the instructional software developed by the participant and not by possible faults in the CBT Generator. Sufficient working of the product was required to proceed to the second measurement. In this second measurement, the quality of the design structure of the products was determined by the experimenters by measuring the number of different types of the instructional, question, and learner-evaluation elements that are used. Also, it was determined whether a fixed instructional sequence (low learner control) or an open sequence (high learner control) was applied, and if a teaser was used to introduce the course in an attractive manner.

Third, the didactical quality of the products was measured in two ways. First, participants were asked to give a *self-score* for the didactical quality of their product on a 10-point scale (1 = 'very bad'; 10 = 'excellent'). In the Netherlands, this is the normal scale for grading learning outcomes and thus very familiar to the participants. Second, using the analytical approach towards determining

didactical quality, three expert raters independently scored all products on their didactical quality. The expert raters were experienced instructional design researchers from TNO Human Factors and not directly involved in this study. They received a short training on the technical possibilities and limitations of the CBT Generator and the embedded didactical model before the expert review, so they would not have unrealistic expectations. For instance, they should not consider alternative user-interface designs whilst the CBT Generator only provides one default interface. The expert raters scored the products in two phases. In the first phase they used a structured checklist reflecting different aspects of the embedded didactical model, such as the implementation of guidelines from the Nine Events of Gagné described above. This structured checklist gave the expert raters a good insight in the didactical quality. The Intra Correlation Coefficient (ICC; Fleiss, 1981) for the *checklist scores* is .72, p = .004, which is good. To directly compare the expert's scores to the participants' self-scores, the expert raters scored the didactical quality of the product in a second phase with one *total score*, on the same 10-point scale utilized by the participants to score the didactical quality. The ICC for the total scores is .63, p = .02. The correlation between the checklist scores and the total scores was high (Spearman's *rho* = .90, p = .000).

Didactical perspective questionnaire. This questionnaire inquired for the didactical perspectives of the participants with regard to learning, instruction, and technology. The design of the questionnaire is based on a classification made by Andriessen and Veerman (2000), who identified a traditional and a progressive design perspective. Participants with a traditional perspective favor 'direct instruction' principles, brought forward by models such as Gagne's Nine Events and Merrill's Component Display Theory (see Reigeluth, 1983). Participants with a progressive perspective favor 'learner-centered' principles, brought forward by models such as problem-based learning and goal-based scenarios.

The questionnaire consisted of seven items. The first five items gave a firm statement on learning, instruction or technology and had to be scored on a 5-point Likert scale (1 = 'disagree'; 5 = 'agree'). Disagreement referred to a traditional perspective and agreement referred to a progressive perspective. The remaining two items were ranking questions. They had each four statements on learning, instruction or technology, which had to be ranked by the participant in their order of importance. If traditional statements were ranked as most important, this indicated a traditional perspective. If progressive statements were ranked as most important, this indicated a progressive perspective. Based on both the scoring and the ranking of the items, each participant was classified as having a traditional, a progressive, or—if no explicit preference for a traditional or progressive perspective could be found—a neutral perspective. The internal consistency of the seven items, expressed as Cronbach's Alpha, was .68.

Development style questionnaire. This questionnaire inquired for the development style of the participants. The design of the questionnaire is based

on a classification of Van Boxtel (2000), who defines three typical development styles: Meaning-directed ('first thinking then acting'), practical-directed ('varying thinking and acting'), and application-directed ('first acting then thinking'). The questionnaire consisted of three items. Each item gave a short description reflecting either a meaning-directed, practical-directed, or application-directed development style. The participants had to score each item on a 5-point Likert scale (1 = 'not fitting me'; 5 = 'completely fitting me') and were then classified on the basis of their average score on the three items. The internal consistency of the three items, expressed as Cronbach's Alpha, was .69.

Evaluation questionnaire. The evaluation questionnaire consisted of 9 statements about the development process, the use of instructional software templates, and the characteristics of the resulting products. Participants expressed their opinion for each statement by scoring them on a 5-point Likert scale (1 = 'totally agree; 5 = 'totally disagree').

Procedure

To prevent initial differences in knowledge of instructional design and ability to work with the CBT Generator, the participants first took part in an eight-day course. The first four-day part focused on Instructional Systems Development (ISD) and creating and interpreting training blueprints. Thus, all participants learned the same structured development approach and the same terminology, preventing the use of different methods and confusion about the assignment in the current study. In the second four-day part of the course, the participants learned to work with the CBT Generator and all of its features, by means of developing a concrete product based on a detailed training blueprint. In this part of the course additional attention was given to interactivity, adaptive learning paths, and 'extra' functions (e.g., glossary, on-line help, teaser). Therefore, familiarity with the tool was guaranteed for all participants.

Immediately after the course, the participants filled out the background questionnaire, the didactical perspective questionnaire, and the development style questionnaire. Then, the development task was given. This task was to be completed in two days (2 x 8 hrs.). The participants worked independently on the task. The experimenter received a list of strict instructions for conducting the experimental procedure and dealing with questions from the participants. Only in the case of technical problems or if a participant got stuck for a long time, did the experimenter offer support. Assistance was provided according to the instructions and strictly limited to technical problems with the tools or the computer, or to conceptual problems with understanding the assignment or the questionnaires. To calculate the effective time spent on the development task, participants were required to register exact start and end times in a time log. The experimenter checked if this was accurately performed. After the development task was finished, the participants filled out the evaluation questionnaire and the experimenter made all products anonymous for the blind expert reviews.

Results

Given the small number of participants, Mann-Whitney tests are used to check for differences between the novice group and the experienced group. With regard to the background variables, no differences between both groups were found with regard to age, job profile, prior education, or experience with educational development projects.

Quantity and Quality of Products

Table 4.1 provides the means and standard deviations of both groups for the quantity of work done, the authoring and technical quality of the developed products, and their didactical quality.

Table 4.1

Means and Standard Deviations from the Product Reviews for the Novice and Experienced Groups

	Novice group (n = 6)		Experienced group (n = 8)	
	M	*SD*	*M*	*SD*
Quantity				
Time on development task (# hours)	8.75	1.86	8.00	1.41
Length of product (# pages)	15.16	9.15	24.75	8.17
# Information elements*	9.50	5.13	18.13	7.22
# Instruction elements	8.00	4.10	13.00	5.04
# Question elements*	1.33	1.75	5.13	3.40
# Learner-evaluation elements	2.17	3.49	1.38	2.07
Authoring and technical quality				
# Instruction elements types (0-12)	2.50	1.05	2.38	.52
# Question elements types (0-12)**	0.83	.98	3.13	1.13
# Learner-evaluation elements types (0-8)	1.33	2.81	0.75	1.17
Open sequence applied (0-1)	.41	.17	.37	.08
Teaser applied (0-1)	.33	.52	.75	.46
Didactical quality				
Participants self-score (1-10)	6.80	.84	5.50	2.43
Expert raters' checklist score (1-10)	2.36	.99	2.89	.43
Expert raters' total score (1-10)	3.33	1.34	4.5	.99

*$p < .05$

**$p < .01$

With regard to quantity, there is no significant difference between the mean number of effective hours that participants in the novice group (M = 8.75) and the experienced group (M = 8.00) spent on the development task, $U(N = 14)$ = 19.50, p = .56. The amount of produced instructional software in number of pages also does not significantly differ between the novice group (M = 15.16) and the experienced group (M = 24.75), $U(N = 14)$ = 12.00, p = .21. There is,

however, a tendency for the experienced group to produce somewhat more software, as indicated by a positive correlation between programming experience and number of produced pages (Kendal's *tau* = .52, *p* = .02).

The average number of information elements used differed significantly between both groups. The novice group used less information elements (*M* = 9.50) than the experienced group (*M* = 18.13), *U*(*N* = 14) = 7.50, *p* = .029. The same applies for question elements, the novice group used less of these elements (*M* = 1.33) than the experienced group (*M* = 5.13), *U*(*N* = 14) = 6.50, *p* = .002. The average number of instructional elements and learner-evaluation elements did not differ significantly between both groups.

With regard to the authoring and technical quality, all products functioned properly and were rated as of 'sufficient' quality. The only significant difference between the novice group and the experienced group was found for the number of different types of question elements: The experienced group used more different types of these elements (*M* = 3.13) than the novice group (*M* = .83), *U*(*N* = 14) = 3.00, *p* = .005. No significant differences were found for the use of different types of instructional- and learner-evaluation elements, nor for applied open sequences or teasers.

With regard to the didactical quality, both the participants and three expert raters gave an overall assessment on a 10-point rating scale. Surprisingly, there is no significant correlation between the participants' self-score and the expert raters' total score, Kendal's *tau* = .14, *p* = .56. As a second measurement, the expert raters gave a score based on a structured checklist. Again, there is no significant correlation between the participants' self-score and the expert raters' checklist score, *tau* = .38, *p* = .12. The self-scores of the novice group (*M* = 6.80) are not significantly different from the self-scores of the experienced group (*M* = 5.50), *U*(*N* = 14) = 9.50, *p* = .29. The expert's total-scores and checklist scores are also not significantly different between the novice group (in order, *M* = 3.33 and *M* = 2.36) and the experienced group (in order, *M* = 4.50 and *U*(*N* = 14) = 11.50, *p* = .11 for the total scores; and *M* = 2.89 and *U*(*N* = 14) = 14.50, *p* = .23 for the checklist scores). Whereas there are no significant differences between the groups, it is noteworthy that novice participants rate the didactical quality of their own products somewhat higher than experienced participants but that the opposite pattern is observed for expert ratings: Experts rate the didactical quality somewhat higher for the experienced group than for the novice group.

Didactical Perspective and Development Style

Table 4.2 presents the distribution of didactical perspectives (traditional, neutral, and progressive) and development styles (meaning-directed, practical-directed, and application-directed) over the participants in the novice group and the experienced group.

Table 4.2
Number of Participants in the Novice Group and the Experienced Group with Particular Didactical Perspectives and Development Styles

	Novice Group (n = 6)	Experienced Group (n = 8)
	Didactical Perspective[a]	
Traditional	3	1
Neutral	2	5
Progressive	0	2
	Development Style	
Meaning-directed	2	1
Practical-directed	4	7
Application-directed	0	0

[a] The data from 1 participant in the Novice Group are missing

The distribution of didactical perspectives does not significantly differ between the novice group and the experienced group ($\chi 2(2, N = 14) = .73$, $p = .69$). A Kruskal-Wallis test shows that there is no significant effect of didactical perspective on participants' self-scores or expert raters' total scores of the didactical quality of the product, the time spent on the development task, and the length of the product.

The distribution of development styles also does not significantly differ between the novice group and the experienced group ($\chi 2(2, N = 14) = .88$, $p = .35$). A Kruskal-Wallis test shows that there is no significant effect of development style on participants' self-scores or expert raters' total scores of the didactical quality of the product, the time spent on the development task, and the length of the product.

Opinions on Templates

Table 4.3 presents the participants' opinions on the development process and the use of the instructional software templates, which they expressed in the evaluation questionnaire. Mann-Whitney tests show no significant differences between the novice group and the experienced group for any of the test items. It appears that the variability within the groups is considerable.

Table 4.3
Results of the Evaluation Questionnaire Indicating Participants' Opinions on the Use of Templates

	Novice group (n = 6)		Experienced group (n = 8)	
	M	*SD*	*M*	*SD*
1. Due to the structure of the templates, I create instructional software of higher didactical quality	3.33	1.03	3.38	1.68
2. By using the templates, the overall quality of my products decreases	3.33	1.03	3.13	1.36
3. Due to the templates, I am able to produce instructional software faster	1.67	.82	3.00	1.77
4. The templates provide me structure	2.83	.75	2.13	1.46
5. The templates provide me more structure than I normally experience	3.67	1.03	3.13	1.88
6. The templates force me too much in a straight-jacket	3.33	1.03	2.13	1.36
7. The number of available types of didactical elements is insufficient	2.00	.89	2.75	.71
8. The 'worked-out example' explains the possibilities of the templates well	1.83	.41	2.00	.93
9. I have used the 'worked-out example' during the development process when something was unclear	4.33	1.03	4.63	.74

Note: All statements could be scored from 1 = 'totally agree', to 5 = 'totally disagree'

Discussion

This study examined the effects of instructional software templates on the development process, the quality of produced instructional software, and the perceived level of support. First, the experienced group was expected to produce more instructional software than the novice group. Indeed, a tendency in this direction was found and compared to the novice group the experienced group filled their pages with more information and question elements. The relatively small difference between the groups was possibly caused by the small amount of time devoted to the development task.

Second, it was expected that both groups would produce final products with a comparable authoring and technical quality. Both groups were, indeed, able to produce working final products and, apart from the use of more question elements by the experienced group, no other differences were found on authoring and technical quality. It seems that the instructional software

templates may effectively compensate for differences in experience with the development of instructional software.

Third, it was expected that the experienced group would produce final products with a higher didactical quality than the novice group, but that the didactical quality of the products from both groups would at least be sufficient. Expert ratings and self-ratings of didactical quality showed, however, no clear differences between the groups. Nevertheless, some measures for authoring and technical quality suggest that the experienced group put somewhat more emphasis on active learning tasks because they used more question elements in their tasks, and varied these learning tasks more often by using different types of question elements. Furthermore, the experts rated the didactical quality a little lower for the novice group than for the experienced group, but participants in the novice group rated the didactical quality of their own final products a little higher than participants in the experienced group. This may possibly be explained by the fact that participants in the novice group performed a new task with a new tool and were, therefore, more likely to be impressed by their final products than the experienced group.

The results are mixed for the overall didactical quality for both groups together. The participants rated the didactical quality of their final products rather positive and their ratings were higher than the expert ratings. The experts, in contrast, were not very positive about the didactical quality: Specifically their checklist scores indicated an insufficient didactical quality. It is not totally impossible that the participants have a better judgment of the didactical quality than the instructional experts (who shared a background in instructional design *research*), because they were more familiar with the subject matter domain and the target groups. However, we are inclined to give a higher weight to the expert ratings because they carefully compared the final products with well-established instructional design principles that are documented in the literature and grounded in educational practices of the Royal Netherlands Army. A possible explanation for the low expert ratings is that most final products had a fixed linear structure, required little active engagement from the learners, used few additional functions, and applied no adaptive branching. The amount of variation in the kinds of learning activities (i.e., different didactical elements) was also rather low. This is despite the fact that before the experimental task, the developers were told that the criteria for the review on product quality would emphasize active learning.

With regard to the perceived level of support, it was expected that both groups would feel they received sufficient pre-structuring support from the templates, resulting in a faster development process and better products. When participants were asked if they could reach a higher didactical quality with the given templates, on the one hand, they reacted rather negative. On the other hand, they claimed the templates did not decrease quality either and were rather positive when asked if they could produce the same products in less time thanks to the templates.

With respect to experienced pre-structuring support, participants indicated they felt that their development process was only moderately structured by the templates. They did not experience this given structure as restrictive. If they felt any limiting effect, it was mainly related to technical changes they could not make, for instance adapting a particular type of didactical element to their wishes. Despite the large amount of already available didactical elements in the templates (33 different elements in total), participants indicated that they would have preferred an even greater choice of didactical elements.

It may be argued that the instructional software templates biased the participants towards the implementation of linear and rather passive didactical models. However, several measures were taken to prevent this. First, the set of available templates actually offered many opportunities to implement varied and highly interactive models. Second, the intensive eight-day course that the participants attended just before the development task covered these opportunities thoroughly and explicitly. And third, during their work on the development task participants were able to consult a fully worked-out example that illustrated these opportunities as well. The participants acknowledged this, although they also indicated that they hardly studied the worked-out example. Taken together the three measures did clearly not yield the desired results. This leads to the general conclusion that the instructional software templates allowed too much freedom for the developers and lacked the necessary amount of *didactical* support. The developers were allowed to express an undesirable bias towards passive, linear instruction based on an information-transmission view, and were allowed to focus on the authoring and technical aspects of their final products at the cost of their didactical qualities. The main beneficial effect of the templates seems to be related to the efficiency instead of the effectiveness of the development process (i.e., mediocre final products with fewer costs rather than superior products with less or equal costs). This is a serious problem because, as argued earlier, didactical quality is the most important characteristic of instructional software.

Future studies should focus on the question of how instructional software templates can help developers create instructional software with a higher didactical quality without restricting their creative freedom. This is particularly important for the development of rich, interactive and flexible learning environments in which meaningful learning tasks are used as the driving force for learning. Lowyck (2001) describes two approaches to support the development of such environments. First, the didactically structured approach directs and supports the developer (a) in applying and filling the information, communication, and interaction components, (b) in determining their didactical functionality, and (c) in creating, configuring, and structuring them accordingly. Instructional software templates will be, thus, fixed in this approach and cannot be changed by the developer. The didactical quality of the final products is mostly determined by the quality of the templates. Second, the open approach only offers a structure to fill in the information, communication, and interaction

components. Instructional software templates may be adapted by the developer, and the didactical quality of the final products is mostly determined by the level of expertise of the developer. The decision to either apply a didactically structured or an open approach will then depend on the instructional paradigm, the characteristics of developers, the available tools and infrastructure, and so forth.

In conclusion, instructional software templates may positively affect the efficiency of the development process and compensate for the developers' lack of experience with the development of instructional software. This building block solution can be highly beneficial for the development of instructional software because more and more people with low instructional design and, particularly, software production skills will become involved. These novice developers often have much (tacit) knowledge of the workplace, subject matter domains, and target groups, as well as good access to a wide range of multimedia materials that can be used to develop instructional software with a high level of authenticity and attractiveness. Even if these developers produce software by means of instructional software templates that do not necessarily enforce the highest didactical quality, the resulting products can be very useful for demonstration and communication of ideas (cf., rapid prototyping, Tripp & Bichelmeyer, 1990). Furthermore, the creation of instructional software templates could be made an integral part of the development process, because it forces designers to reflect on their work and to make their didactical perspectives on learning and instruction explicit. In turn, "best practices" of (experienced) developers with regard tot authoring, technical, and didactical issues can be captured in a *design-patterns language*. As described above, design patterns are standard solutions to common problems, each capturing the essence of a particular practice. Although each of these patterns can be simple, from a set of design patterns that work together to generate complex behavior and complex artifacts, a design pattern language can arise (see for example http://www.pedagogicalpatterns.org). Such a language can feed the creation of new templates that connect better to the needs, abilities, and limitations of developers.

References

Andriessen, A., & Veerman, A. L. (2000). Samenwerkend telestuderen in het universitair onderwijs. In J. van der Linden & E. Roelofs (Eds.), *Pedagogische studiën: Leren in dialoog* [Pedagogical Studies: Learning in Dialogue] (pp. 157-179). Groningen, The Netherlands: Wolters-Noordhoff.

Boot, E. W., & Bots, M. (2002). Learning object creation, management and reuse by non-experienced content developers. *Proceedings of the I/ITSEC 2002* (pp. 446-453). Orlando, FL.

Boot, E. W., & van Rooij, J. C. G. M. (1999). *Gestructureerde ontwikkeling van Computer-Ondersteund Onderwijs met behulp van templates: Analyse* [Structured development of Computer Based Training with templates: Analysis] (Report TM-99-A068). Soesterberg, The Netherlands: TNO Human Factors Research Institute.

Chapman, B. (2003). *LCMS report: Comparative analysis of enterprise learning content management systems*. Sunnyvale, CA: Brandon-Hall.com.

Cline, R. W., & Merrill, M. D. (1995). Automated instructional design via instructional transactions. In R. D. Tennyson & A. E. Barron (Eds.), *Automating instructional design: Computer-based development and delivery tools* (pp. 317-353). New York: Springer.

De Hoog, R., Kabel, S., Barnard, Y., Boy, G., DeLuca, P., Desmoulins, C., Riemersma, J., & Verstegen, D. (2002). Re-using technical manuals for instruction: Creating instructional material with the tools of the IMAT project. *ITS 2002 Conference - Integrating Technical and Training Documentation Workshop*. San Sebastian, Spain.

De Jong, T., Limbach, R., Gellevij, M., Kuyper, M., Pieters, J., & van Joolingen, W. R. (1999). Cognitive tools to support the instructional design of simulation-based discovery learning environments: The SimQuest authoring system. In J. van den Akker, R. M. Branch, K. Gustafson, N. Nieveen, & Tj. Plomp (Eds.), *Design approaches and tools in education and training* (pp. 215-225). Dordrecht, The Netherlands: Kluwer.

Fleiss, J. L. (1981) *Statistical methods for rates and proportions* (2nd Ed.). New York: Wiley.

Gagné, R. M., & Briggs, L. J. (1979). *Instructional technology: Foundations*. Hillsdale, NJ: Lawrence Erlbaum.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns*. Reading, MA: Addison-Wesley.

Gettman, D., McNelly, T., & Muraida, D. (1999). The guided approach to instructional design advising (GAIDA): A case-based approach to developing instructional design expertise. In J. van den Akker, R. M. Branch, K. Gustafson, N. Nieveen, & Tj. Plomp (Eds.), *Design approaches and tools in education and training* (pp. 175-182). Dordrecht, The Netherlands: Kluwer.

Gibbons, A. S., Nelson, J., & Richards, R. (2001). The nature and origin of instructional objects. In D. A. Wiley (Ed.), *The instructional use of learning objects* (pp. 25-58). Bloomington, IN: Association for Educational Communications and Technology.

Goodyear, P. (1997) Instructional design environments: Methods and tools for the design of complex instructional systems. In S. Dijkstra & N. Seel (Eds.), *Instructional design: International perspectives* (pp. 83-111). Hillsdale, NJ: Lawrence Erlbaum.

Kolodner, J. L. (1997). Educational implications of analogy, a view from case-based reasoning. *American Psychologist*, *52*, 57-66.

Leshin, C. B., Pollock, J., & Reigeluth, C. (1992). *Instructional design strategies and tactics.* Englewood Cliffs, NJ: Educational Technology Publications.

Locatis, C., & Al-Nuaim, H. (1999). *Interactive technology and authoring tools: A historical review and analysis.* Educational Technology, Research and Development, 47, *63-76.*

Lowyck, J. (2000), Van personal computer tot E-platform, implicaties voor een ontwerpkunde. In R. Koper, J. Lowyck, & W. Jochems. *Van verandering naar vernieuwing* [From change to renewal] (pp. 53-71). Heerlen, The Netherlands: Open University of the Netherlands.

Mayer, R. E. (2001). *Multimedia learning*. Cambridge, UK: Cambridge University Press.

Merrill, M. D. (2002). First principles of instruction. *Educational Technology, Research and Development*, *50*, 43-59.

Merrill, M. D., & ID2 Research Group (1998). ID expert: A second generation instructional development system. *Instructional Science, 26*(3-4), 234-262.

Merrill, M. D. (1997). Learning-oriented instructional development tools. *Performance Improvement, 36*(3), 51-55.

Nantel, R. (2003). *Authoring tools 2004: A buyer's guide to the best e-learning content development applications* (executive summary). Retrieved February 12, 2004, from http://www.brandon-hall.com

Reigeluth, C. M. (Ed.). (1983). *Instructional design theories and Models*. Hillsdale, NJ: Lawrence Erlbaum.

Rosenberg, M. J. (2000). *E-learning, strategies for delivering knowledge in the digital age*. New York: McGraw-Hill.

Simons, R. J. (2002). *Digitale didactiek: Hoe (kunnen) academici leren ICT te gebruiken in hun onderwijs* [Digital learning and instruction: How academics (can) learn to use ICT in their education]. Inaugural address. Utrecht, The Netherlands: University of Utrecht.

Spector, J., & Muraida, D. (1997). Automating design instruction. In S. Dijkstra, N. Seel, F. Schott, & D. Tennyson (Eds.), *Instructional design: International perspectives* (Vol. 2) (pp. 59-81). Mahwah, NJ: Lawrence Erlbaum.

Sweller, J., van Merriënboer, J. J. G., & Paas, F. (1998). Cognitive architecture and instructional design. *Educational Psychology Review*, *10*, 251-296.

Tennyson, R. D., & Baron, A. E. (Eds.). (1995). *Automating instructional design: Computer-based development and delivery tools.* New York: Springer-Verlag.

Tripp, S., & Bichelmeyer, B. (1990). Rapid prototyping: An alternative instructional design strategy. *Educational Technology, Research and Development, 38*, 31-44.

Van Merriënboer, J. J. G. (2002). De ontbrekende didactiek van E-leren [The missing didactics of e-learning]. *Pedagogische Studieën*, *79*, 494-502.

Van Merriënboer, J. J. G., & Martens, R. (2002). Computer-based tools for instructional design. *Educational Technology, Research and Development*, *50*, 5-9.

WBTIC (2004). *Overview of tools to design, author, deliver, and manage online learning*. Retrieved March 2, 2004, from http://www.wbtic.com/resources_tools.aspx?au=1

Wiley, D. A. (Ed.). (2001). *The instructional use of learning objects*. Bloomington, IN: Association for Educational Communications and Technology.

Chapter 5

Solutions for Developing Instructional Software by Creating and Reusing Learning Objects[4]

Abstract

Two studies test the potential of template, automation, and intermediate product solutions to overcome problems that hamper the efficient development of instructional software by reuse of learning objects. In the first study, the templates and automation solutions were applied by developers (N = 8) who created and reused large as well as smaller multimedia learning objects—in a familiar and an unfamiliar domain. Developers judged both solutions positively, and rated working with didactical meaningful learning objects higher than working with multimedia objects. However, no differences between the familiar and unfamiliar domain were found and the developers made several remarks on the limitations of reuse. In the second study, the automation solution in combination with a set of (a) regular templates, (b) extended templates, and (c) intermediate products were applied by developers (N = 15) who created and reused learning objects. As expected, the automation solution in combination with the intermediate products yielded the highest quality learning objects, followed by the extended templates and, finally, the regular templates. The two studies show that there is no single solution for all problems of reuse: The problems will only be solved if a well-chosen combination of solutions is applied.

[4] Boot, E., & van Merriënboer, J.J.G. (submitted). *Solutions for Developing Instructional Software by Creating and Reusing Learning Objects*

Reuse of learning objects (LOs)—digital units of information with an instructional purpose—is believed to promote the efficient development of instructional software. Two kinds of LOs can be identified in this modular approach. First, didactically meaningful LOs are relatively large units of learning material (e.g., Computer-Based Training modules or web pages) that contain interaction possibilities such as questions and tests that can be tracked by e-learning systems. Second, multimedia LOs are smaller components such as text documents, pictures, and interactive animations that contain interaction possibilities that are not tracked by e-learning systems. Both kinds of LOs are created, centrally stored, retrievable, and applicable for multiple purposes (i.e., "once made - used many"). Learning-technology standards provide the common frameworks to technically enable this type of reuse.

The need for reuse is currently increasing, as modern task-directed and competency-based teaching models promote the development of powerful electronic learning environments. Such environments are multimedia-rich, requiring many multimedia LOs. Also, they often include adaptive learning trajectories, requiring multiple sets of didactically meaningful LOs to tailor the instruction towards the needs of individual learners. To save development costs, novice developers with low production experience, such as subject matter experts, teachers, and instructional designers, are often involved in the process of reusing LOs. However, the many complex didactical and technical issues potentially exclude such inexperienced developers from a smooth reuse process. Mostly, they are only involved in the development of instructional software for a short period of their career, which makes it difficult to gain the necessary experience. An important question is how to assist these developers by means of new support solutions. In this article, first, five problems of reuse are presented as well as three possible solutions to overcome these problems. Subsequently, two studies are described that empirically test the effectiveness of different combinations of those solutions. The article ends with a general discussion, presenting the conclusions as well as the theoretical and practical implications of the presented studies.

Problems of Reuse

Van Merriënboer and Boot (2005) identify five problems of reuse: The metadata problem, the arrangement problem, the exchange problem, the context problem, and the pedagogical function problem.

The Metadata Problem

This problem refers to the fact that it is difficult and extremely labor intensive to specify metadata for large sets of LOs. Metadata is information to "label" LOs in order to enable an efficient search for them in databases. Examples of metadata are the title, the producer, the possible application, the content, the size, and so forth. There is a lively discussion on the number of necessary metadata fields. If a developer of an object fills out too few fields,

other developers, searching for objects, will probably be overwhelmed with a large amount of possibly relevant LOs. However, using more fields heavily increases the workload associated with the specification of LOs. And while it may help other developers to find exactly what they want, it reduces the chance that they find anything at all because the chance that an object has the features a, b, and c is smaller than the chance that it has only the features a and b. Furthermore, there is also discussion on the nature of the metadata. For instance, well-defined metadata fields make it difficult or even impossible for an individual developer to express his intentions unambiguously, while loosely defined fields yield communication problems between developers and, eventually, between e-learning systems.

The Arrangement Problem

This problem refers to the fact that combining and sequencing LOs into larger arrangements is not always easy and self-evident. This can be illustrated by the LEGO© metaphor. Basic LEGO© bricks are only appropriate to build simple structures. For building more complex structures, one requires the equivalent of more advanced LEGO© bricks, as found in Technical LEGO© (e.g., axles, gearwheels, receptors, programmable bricks etc.). In contrast to basic LEGO© bricks, Technical LEGO© elements differ in their external structures (i.e., the way they can be attached to each other) and thus cannot be combined with every other element. Also, they differ in their internal structures (i.e., the function of the element, like an axle or gearwheel) so that they can only be combined into certain arrangements to form bigger elements. This metaphor illustrates that differences between LOs can prevent valid arrangements. For instance, if two LOs differ in their external structures because they yield incommensurable assessment information (e.g., one is using American A-B-C grading, the other the European 10-point scale), they cannot easily be combined into one valid arrangement. As an example for difference in internal structure, two LOs (e.g., an annotated picture and a piece of text) may together yield an invalid arrangement because in one LO another word is used to refer to the same thing as in the other LO (e.g., the word "screen" is used in the annotated figure and the word "monitor" is used in the piece of text), which will make the arrangement highly confusing for the learner. With regard to the arrangement problem, it may even be argued that only large instructional arrangements like complete lessons or courses can be reused effectively (Wiley, 2000).

The Exchange Problem

This problem refers to the fact that it may be difficult to exchange LOs between developers and e-learning systems. From a psychological viewpoint, the readiness to share LOs is not self-evident, as this implies access to personal notions and ideas by others. This raises a psychological threshold for developers to let others reuse their materials. But the other side of the coin is that developers do not easily accept objects that were not developed by them, something known as the "not-invented-here" syndrome. Second, organizational

factors like security policies (e.g., for military information) and infrastructure (e.g., firewalls that prevent using plug-ins or java applets) may prohibit an effective exchange of LOs. Third, current regulations concerning Intellectual Property Rights (IPRs) often limit the sharing of LOs.

The Context Problem

This problem refers to the fact that effective LOs cannot be created in isolation without an implicit or explicit instructional setting, target group, or other contextual descriptors. This may seriously hinder reuse. Suppose, for example, that a LO is created consisting of a picture of a piece of machinery, one part of the machine that is highlighted by color-coding, and an explanatory text for this highlighted part. This LO can be effectively used for a presentation, but not for a test because the explanatory text may not be used as feedback (i.e., another pedagogical purpose); it can be effectively used for individual e-learning but not for teacher-led instruction because the explanatory text may be redundant with the explanation of the teacher and deteriorate learners' performance (i.e., another instructional setting causing the so-called "redundancy effect"; see van Merriënboer & Sweller, 2005), and it can be effectively used for most learners but not for the color-blind because of the color-coding (i.e., another target group). In sum, this implies that every LO has its own context-specificity, which makes it hard to apply it in a context other than that for which it was originally created.

The Pedagogical Function Problem

This problem refers to the fact that it is difficult to express the pedagogical intentions of a LO by means of technical properties such as metadata, leading to sub-optimal reuse. Properties like size or format can be sufficiently described in metadata, but the pedagogical function is extremely hard to specify. First, this is caused by the fact that LOs can often fulfill different functions. For instance, according to Merrill's Component Display Theory (1983) a photograph of an eagle can be used (1) as an example or instance of the concept "bird"; (2) as a test item where learners must classify the bird as an eagle; (3) as an alternate representation of a textual description of an eagle, and so on. Furthermore, the pedagogical function of one LO may require other complimentary LOs, which may or may not be available to the developer. For instance, the pedagogical function of the photograph of the eagle may be that of an example, but this requires the joint availability of another LO with the pedagogical function of a generality (i.e., a definition of the concept "bird").

Solutions for Reuse

Van Merriënboer and Boot (2005) proposed three solutions to overcome the problems of reuse: Templates instead of instantiations, technically automate what can be automated, and intermediate instead of final products.

Templates Instead of Instantiations

Developers often make ample use of implicit or explicit templates: Pre-structured "moulds" of instructional software. These can be used (a) to organize lessons (e.g., presenting content–providing practice with feedback—discussing results, etc.); (b) to reach particular types of instructional objectives (e.g., to support learning a procedure, state the general steps of procedure—provide several demonstrations—require learners to perform the procedure, and so on in accordance with Component Display Theory; see Merrill, 1983), and (c) to design computer screens of the lessons. A focus on templates rather than instantiations may increase the effectiveness of reuse. A similar orientation on templates can be found in Object Oriented Programming (OOP), which offers a solution for dealing with highly complex development processes by decomposing the software into independent units that can be easily reused because of the level of abstraction provided (Booch, 1994).

Templates diminish the arrangement problem because they offer better opportunities than instantiations to make valid combinations of LOs. For instance, two instantiations of which one uses the American A-B-C grading system and the other uses the European 10-point grading system are difficult to combine in one arrangement. If two templates were used rather than two instantiations, offering the opportunity to specify the required grading system (e.g., by selecting it from a list of possible options), there would be no arrangement problem because one could simply specify the same grading system for each LO. Furthermore, the exchange problem and, in particular, the "not-invented-here" syndrome are at least partly solved because developers are expected to specify the instantiations according to their own preferences. Such specification will help to develop a sense of ownership. Templates partly solve the context problem because the context-sensitive information needs not be in the templates but only in the instantiation of this template. Templates offer the developer the opportunity to specify the context-sensitive information. For instance, if a developer is using a template for reaching an instructional objective of the type "using a procedure" (i.e., give general steps of the procedure, give several demonstrations, ask learner for applications), the developer may specify a demonstration that is explained in English for one context or target group, and a demonstration that is explained in another language for another context or target group. Summarizing, a focus on templates instead of instantiations may help to solve the arrangement, exchange, and context problem. However, an important implication is that templates should contain as little contextual information as possible so that the developer can precisely specify this context-specific information.

Technically Automate What Can be Automated

Advancements in information technology may also offer facilities that support reuse of instructional materials. Automation is especially beneficial for processing large amounts of information. The best example pertains to the problem of metadata creation and exchange. Current research (Boot & Veerman,

2003) aims at the development of algorithms for the automatic analysis of multimedia content and the semantic indexing of this content in metadata fields. Such automation may not only be more cost-effective but also yield more objective metadata than indexing by hand. Even if complete automation is not feasible, the specification of metadata and the search for LOs on the basis of metadata can be strongly supported with automated tools. For instance, objective features of LOs, such as format, size, and language can automatically be generated and added to the metadata specification of that LO. The same applies for objective features of the developer, such as his name, organization and IPRs, and for information from formal vocabularies, such as classifications (e.g., a scalpel is an instance of the category of medical instruments) and particular keywords (e.g., Healthcare, Surgery). Finally, search and retrieval features of e-learning systems can provide intuitive user interfaces that make it easier to use these types of metadata.

Intermediate instead of Final Products

While LOs are generally defined as "digital units of information with an instructional purpose", they are typically limited to the final products that can be directly presented to learners. Of course, there are many other informational units with an instructional purpose, such as (a) the results of a contextual analysis, target group analysis, or task analysis; (b) descriptions of performance and instructional objectives for a particular lesson or course; (c) blueprints including learning activities providing the basis for the development of instructional materials, and so forth. These intermediate products contain rich information that describes the final products for which they were made, but they are suitable for reuse as well.

Intermediate products may help solve the arrangement problem because they provide insight into valid arrangements and so guide the selection and sequencing of final products. For example, if the results of a task analysis (i.e., intermediate product) provide an ordered sequence of decision steps, this facilitates the search and arrangement of demonstrations (i.e., final products) for each step. And if a list of instructional objectives (i.e., intermediate product) is available for a new educational program, this facilitates the search and arrangement of courses (i.e., final products) that constitute the program. Intermediate products may also help to solve the context problem because they provide rich information about the final products, which facilitates the finding of LOs that fit a new context. Actually, the intermediate products can fulfill the same role as the metadata specified for final products, but they are expected to be more effective because they provide more content-related information. For instance, if a developer is searching for a picture of a particular type of milling machine, the result of a task analysis on milling (i.e., intermediate product) will be very helpful for finding the most effective picture (i.e., final product) because it indicates the controls and displays that need to be operated by the learners and, therefore, should be visible in the picture. Typically, this type of information (i.e., which displays and controls are visible on the machine) is not

part of the picture's metadata. Furthermore, intermediate products may also help to solve the pedagogical function problem because they provide rich information that helps the developer to determine which pedagogical functions can or cannot be fulfilled by a final product. For instance, in our example it is possible that the results from the task analysis allow the developer to determine that all controls and displays—necessary to operate the milling machine—are visible on the picture. The developer may then decide to use this picture not as an illustration but as a test, in which the learner must describe the steps and simultaneously point out the necessary controls and displays for operating the machine.

An important condition for the intermediate product solution is that developers carefully document these intermediate products in a digital form, preferably in databases that interrelate intermediate and final products. Computer-based instructional design tools may help to do so (for an example of such a tool based on the 4C/ID-model, see de Croock, Paas, Schlanbusch, & van Merriënboer, 2002, and www.enovateas.com).

Testing Possible Solutions for Problems with Reuse

The three presented solutions (templates, automation, intermediate products) and, especially, combinations of the three are believed to support developers to solve the five presented problems of reuse. However, there is no empirical evidence yet that they will actually provide support. Therefore, two studies are conducted to test if different combinations of these solutions, as embedded in computer-based tools for creating and reusing LOs, really support developers. These studies are set up in an authentic setting to investigate how implementations of the proposed solutions are perceived by the developers (Study 1) and how they affect the quality of created LOs (Study 2).

The first study focuses on a combination of the template and automation solutions. Inexperienced developers create and reuse didactically meaningful LOs and multimedia LOs in a familiar and an unfamiliar subject matter domain. In general, it is expected that developers will positively perceive the support provided by the combined templates and automation solution and the quality of their created LOs. A first hypothesis is that developers more positively perceive the support when they are working with didactically meaningful LOs than when they are working with multimedia LOs. This is true because metadata become increasingly important when you search semantically rich elements, which must be assessed on their potential pedagogical application and their possible combinations with other LOs. A second hypothesis is that developers more positively perceive the provided support when they are working in an unfamiliar domain compared to a familiar domain, because an unfamiliar domain forces them to rely more on metadata.

The second study focuses on the effectiveness of combinations of the template, automation, and intermediate product solutions. In a familiar domain, inexperienced developers create and reuse LOs that take the form of cases, that

is, learning tasks that describe an authentic problem situation and ask learners for possible solutions (Schank, Berman, & MacPherson, 1999). Three conditions are distinguished and compared with regard to the quality of created cases. The condition that combines the "regular" template and automation solutions serves as a baseline and is comparable to the combined solution applied in Study 1. First, it is hypothesized that a combined solution with *extended* templates and automation results in a higher quality of created cases than the baseline condition because the extended templates contain job aids that provide meaningful information on how to use the templates. Second, it is hypothesized that an integrated solution with regular templates, automation, as well as intermediate products also results in higher quality of created cases than the baseline condition because the intermediate products provide meaningful information on how to use the LOs.

Study 1: Perceptions of the Automation and Template Solutions

Method

Participants

Eight developers of—paper-based and computer-based—educational materials from the Royal Netherlands Army (RNA) participated in this study. They had a background in a military domain (engineering, medical support, artillery, or air defense) and worked at one of the Educational Development Centers of the RNA. The participants worked in four teams of two persons each, who were both employed at the same Educational Development Center.

Materials

Development tools. The Integrated Development Environment (IDE) used by the participants consisted of (a) the Sharable Content Object (SCO) Generator, and (b) a Repository. The SCO Generator provided the template solution for this study, by offering pre-structured support on (a) didactical, (b) authoring, and (c) technical aspects of developing LOs. With respect to the didactical aspects, the templates provided a number of default instructional design structures that are familiar to the participants, such as Gagné's Nine Events (1965), presentation-practice-feedback sequences, and simple-to-complex orderings of learning tasks (see, e.g., Leshin, Pollock, & Reigeluth, 1992). With respect to the authoring aspects, the templates provided a number of empty software structures that could easily be instantiated with multimedia materials to become LOs. Aspects such as timing of feedback, tracking and tracing of learning results, interaction design, and navigation structures could easily be (re)configured. With respect to the technical aspects, the templates provided a number of structures for "packaging" LOs, that is, storing them according to the specifications of ADL-SCORM (2004). A distinction is made between SCOs and Assets. The first are didactically meaningful objects provided with metadata, which can be executed by a learning management system (e.g., modules, lessons, learning tasks). The second are multimedia objects provided with

metadata, which are used to build an SCO and cannot be executed themselves (e.g., graphics, text documents, video and audio clips).

The *Repository* is a database that supports the storage and retrieval of LOs. It provides the automation solution for this study and is based upon the Teletop learning management system (Strijker, 2004). The process of *storing* LOs is largely automated because the participant (1) can upload a new LO to the database by operating a simple user interface, and (2) needs to specify only a selected, limited set of metadata fields suggested by the system. Metadata that can be automatically generated are specified by the system, which analyzes the objective features of the LO (e.g., size, language) and the user-profile (e.g., name and organization of the author).

Table 5.1
An Overview of the Used Metadata Fields with Typical Examples Specified

Category	Metadata Field	Examples of specified metadata
General	Title	"Theory on Medical Instruments"
	Description	"Lesson on the theory of Medical Instruments"
	Keywords	"Medical Instruments", "Scalpels", "Pliers"
Technical	File format	"Word document"
Classification	Purpose	"Medical Discipline"
	Description	"NL", "Army", "Logistics", "Healthcare", "Medical Services", "Operational Healthcare", "Medical Treatment"
	Keyword	"Medical Instruments"

Table 5.1 provides an overview of the used metadata fields, based on the specifications of ADL-SCORM. The process of *retrieving* LOs is partly automated through advanced search methods. A search on metadata fields yields an overview of all applicable LOs. A search on icons yields a "thumbnail" overview of all graphical files. A search on operators offers an overview of LOs that contain particular keywords and/or are created in a particular time frame.

Tasks. The development tasks were performed in two sessions of three days each. In both Session 1 and Session 2, five assignments were conducted by all four teams:

1. Storing Assets that were pre-installed on participants' computers, into the Repository. This assignment took 4 hours and was used to test the automation solution.
2. Retrieving Assets that were stored by one of the other teams as part of Assignment 1, from the Repository. This assignment also took 4 hours and was used to test the automation solution.
3. Developing learning content with the SCO generator, based upon a small training blueprint, describing the learning objectives and the training

context. The learning content was created from the SCOs built by the Assets retrieved as part of Assignment 2. Sometimes, multiple SCOs had to be reconfigured to make valid arrangements. This assignment took 8 hours and was used to test the template solution.

4. Storing SCOs that were created as part of Assignment 3, into the Repository. This assignment took 4 hours and was used to test the automation solution.
5. Retrieving SCOs that were stored by one of the other teams as part of Assignment 4, from the Repository. This assignment took 4 hours and was used to test the automation solution.

Session 1 and 2 differed from each other with regard to the *familiarity* of the participants with the subject matter domain. Session 1 dealt with medical instruments and field orientation, topics that were unfamiliar to the participants. Session 2 dealt with ammunition awareness, medical materials, tank recognition, and aircraft recognition—topics the members of each team were specialized in. This allowed for testing the hypothesis that the template and automation solutions are rated more useful for unfamiliar domains than for familiar domains, because in the unfamiliar domain the participants are forced to rely more on metadata.

Experimental rooms. The four teams worked independently of each other, in rooms equipped with one computer for each participant. Each computer had the SCO Generator installed and broadband LAN access to the Repository which was installed on a central server. The Assets for the first assignment of each session were available in a map presented on the desktop of each computer. The Repository was filled with more than 1000 Assets in a variety of file formats in order to represent a realistic working situation.

Measurements

Background questionnaire. This questionnaire collected information on participants' (a) experience with developing educational materials including the production of instructional software, (b) previous coursework, and (c) experience with reusing multimedia components and parts of lessons. Participants were asked with an open question about their current experiences with regard to reuse.

Monitoring. In a log file, the Repository automatically recorded the number of stored and retrieved LOs and the related time in minutes.

Evaluation questionnaire. The questionnaire consisted of two parts. The first part contained 10 questions pertaining to particular aspects of the SCO Generator and the template solution. For questions 1-5, the participants used a 5-point Likert scale; for question 6, they used a "yes-no" scale, and for questions 7-10, they used a 10-point scale (1 = "very bad", 6 = "satisfactory", 10 = "excellent") (see Table 5.2). The second part contained 8 questions pertaining to the automation solution. The participants rated particular aspects on a 10-point scale for both SCOs and Assets, to enable the comparison between didactically oriented LOs and multimedia LOs (see bottom part of Table 5.3). This allowed

for testing the hypothesis that the automation solution is rated as more useful for SCOs than for Assets. Participants were asked to write down any remarks they had with regard to the assignments, the process of reusing LOs, and the development process in general.

Product reviews. The experimenter rated the technical functioning of all products as sufficient or insufficient.

Procedure

All participants took part in a one-day training session. The purpose and procedure of the study were explained and the SCO Generator and Repository were demonstrated. The purpose of LOs and the use of metadata fields were discussed, without discussion about which values should be used to fill out the fields. A worked-out example of a course created with the SCO Generator was shown. A paper-based job-aid with instructions for how to use the Repository was made available. The participants filled out the background questionnaire before Session 1. In both sessions, the four teams could *not* consult with each other, forcing participants to rely on available metadata specified by other teams, rather than personal communication. An experimenter was always present to answer questions, but this assistance was limited to technical problems and difficulties with understanding the assignments. The participants filled out the evaluation questionnaire after each session to enable a comparison between an unfamiliar domain (Session 1) and a familiar domain (Session 2). Finally, the experimenter reviewed all products.

Results

Participants mean development experience (design and production activities) was 4.75 years (*SD* = 3.01); they completed an average of 3.87 instructional software products (*SD* = 3.52), and they took an average of 2.62 courses on the development of educational materials (*SD* = 1.41). They rated their experience with template-based development tools as "below average" (*M* = 1.75, *SD* = 0.71 on a 5-point Likert scale on which 1 = not experienced and 5 = very experienced) and their experience with the CBT generator, a predecessor of the SCO Generator, as "average" (*M* = 2.50, *SD* = 1.07). The Repository was completely new to them. Seven participants had experience with reusing Assets—four participants reused assets from only one course and three participants reused assets from multiple courses. Four participants had experience with reusing SCOs—two participants reused SCOs from only one course and two participants reused SCOs from multiple courses.

From the participant's current experience of reuse, four central issues appeared. First, all participants think that there is much reusable learning content available in their organization. Second, they often find it difficult to find relevant learning content. Third, they find it difficult and laborious to adapt learning content from other resources for their own use. Fourth, they also find it difficult and laborious to create metadata files and specify relevant metadata.

Template Solution

The participants developed roughly the same number of SCOs in the unfamiliar domain (Session 1, *M* = 12.38, *SD* = 5.55) and the familiar domain (Session 2; *M* = 13.50, *SD* = 7.87). Table 5.2 presents an overview of the means and standard deviations of participants' ratings of specific aspects of the SCO Generator and the template solution.

Table 5.2
Study 1: Participant Ratings for the SCO Generator and Template Solution

	Unfamiliar domain (Session 1)		Familiar Domain (Session 2)	
	M	*SD*	*M*	*SD*
1. Quality of user interface of SCO generator [a]	3.50	.93	3.63	.74
2. Quality of structure of SCO generator	2.37	1.51	2.25	1.38
3. Quality of information presentation by SCO generator	1.75	1.04	3.50	1.41
4. Quality of didactical templates in SCO generator	3.13	1.55	3.13	.64
5. Possibility to apply own ideas	1.88	.99	2.37	1.18
6. Desire to have influence on reuse of others by own materials [b]	.75	.46	.43	.53
7. Satisfaction with end result (i.e., the created reusable LOs)	5.87	1.88	5.13	2.58
8. Expectation of availability of reusable material in RNA	6.50	1.77	6.17	1.72
9. Usability of SCO Generator for developing reusable LOs	5.75	1.38	5.88	1.46
10. Suitability of complete *template solution* for developing reusable LOs	5.75	1.67	5.75	1.48

[a] Questions 1-5 are rated on a 5-point scale (1 = "very bad"; 5 = "very good").
[b] Rated as "no" (0) or" yes" (1).
[c] Questions 7-10 are rated on a 10-point scale (1 = "very bad", 6 = "satisfactory", 10 = "excellent").

Participants rated the quality of information presentation by the SCO Generator (question 3) lower when working with unfamiliar content (*M* = 1.75, *SD* = 1.04) than when working with familiar content (*M* = 3.50, *SD* = 1.41; *Wilcoxon Signed Rank test: W* = 0.00, $p < .05$. No other significant differences between the unfamiliar and familiar domain were found.

The quality of the interface (question 1) and the didactical templates (question 4) were rated above satisfactory, and the participants expected a high availability of reusable materials in the RNA (question 8). The rating on the

desire to have influence on the reuse of self-developed materials by others was between "yes" and "no" (question 6). Participants rated their satisfaction with the end result (question 7), their perceived usability of the SCO Generator (question 9), and the suitability of the complete template solution (question 10) as almost satisfactory. Finally, the quality of the structure (question 2) and the possibility to apply their own ideas (question 5) were rated below satisfactory.

The product review indicated that all teams created working LOs with the SCO Generator. There were some minor technical problems with three of the products due to bugs in the SCO Generator.

Automation Solution

All participants were able to store and retrieve LOs. Table 5.3 presents an overview of the means and standard deviations of the times for storing and retrieving LOs, the number of retrieved LOs, and the participants' ratings on eight aspects of the Repository and the automation solution.

Table 5.3
Study 1: Participant Ratings for the Repository and Automation Solution

	Unfamiliar content (Session 1)				Familiar content (Session 2)				Total (Session 1 plus 2)			
	SCOs		Assets		SCOs		Assets		SCOs		Assets	
	M	*SD*	*M*	*SD*	*M*	*SD*	*M*	*SD*	*M*	*SD*	*M*	*SD*
Average time for storing LOs (minutes) (assignments 1 and 4)	4.25	3.41	3.87	1.88	2.75	1.98	7.43	10.36	3.50	2.22	5.57	6.00
Average time for retrieving LOs (assignments 2 and 5)	2.50	.93	4.00	3.29	1.50	1.07	2.75	1.16	2.00	0.53	3.37	1.48
Average number of retrieved LOs (assignments 2 and 5)	20.25	17.36	27.13	11.23	14.13	7.06	28.00	8.83	17.18	9.15	27.58	8.89
Ratings on aspects of the Repository and automation solutions 10-point scale from 1 (very bad) to 10 (excellent)												
1. Relevance of metadata set	7.25	1.04	6.37	1.18	6.75	.71	7.00	.75	7.00	.53	6.68	.65
2. Unequivocalness of metadata set	6.83	.75	5.63	2.26	6.62	.74	5.63	1.51	6.83	.68	5.63	1.15
3. Usefulness of metadata set	6.75	.71	6.37	.52	6.25	1.38	6.63	.52	6.50	.75	6.50	.37
4. Usability of the specified metadata for others	7.28	.95	6.13	2.03	7.14	.89	7.00	.82	7.00	0.54	6.28	.64
5. Usability of the Repository for storing LOs	6.87	.99	6.43	1.27	6.71	.75	6.50	.93	6.64	0.55	6.50	1.00
6. Usability of the Repository for retrieving LOs	7.00	.82	6.00	1.07	6.25	1.48	6.25	.88	6.86	.69	6.13	.64
7. The complete automation solution for storing LOs	7.25	1.16	4.87	1.73	6.63	.74	6.25	.71	6.94	1.25	5.56	.94
8. The complete automation solution for retrieving LOs	6.75	.71	5.75	1.28	6.50	.75	6.63	1.06	6.25	.58	6.18	.65

As indicated in the top of Table 5.3, participants spent between 2.75 and 7.43 minutes for storing a new LO, between 1.50 and 4.00 minutes for retrieving a LO, and retrieved a minimum of 14 and a maximum of 28 LOs. The following significant differences are found by Wilcoxon Signed Rank tests. When working with familiar content, participants retrieved more Assets (M = 28.00, SD = 8.83) than SCOs (M = 14.13, SD = 7.06; W = 0.00, p < .05). Also, over both sessions, participants retrieved more Assets (M = 27.58, SD = 8.89) than SCOs (M = 17.18, SD = 9.15; W = 3.00, p < .05). Over both sessions, participants needed less time to retrieve SCOs (M = 2.00, SD = .53) than Assets (M = 3.37, SD = 1.48; W = 4.00, p < .05).

On eight occasions, SCOs were rated higher than Assets on one particular aspect. The relevance of the metadata set (Question 1) was rated 7.25 for SCOs and 6.37 for Assets (W = 6.00, p < 0.5) in Session 1. Unequivocalness of the metadata set (Question 2) was rated 6.62 for SCOs and 5.63 for Assets in Session 2 (W = 0.00, p < .05). The usability of the specified metadata for others (Question 4) was rated 7.28 for SCOs and 6.13 for Assets (W = 2.00, p < 0.5) in Session 1, and rated 7.00 for SCOs and 6.28 for Assets (W = 0.00, p < 0.5) over both sessions. The usability of the Repository for retrieving LOs (Question 6) was rated 6.86 for SCOs and 6.13 for Assets (W = 0.00, p < 0.5) over both sessions. The complete automation solution for storing LOs (Question 7) was rated 7.25 for SCOs and 4.87 for Assets (W = 0.00, p < 0.5) in Session 1, and rated 6.94 for SCOs and 5.56 for Assets (W = 0.00, p < .05) over both sessions. Finally, the complete automation solution for retrieving LOs (Question 8) was rated 6.75 for SCOs and 5.75 for Assets (W = 0.00, p < 0.5) in Session 1.

With regard to familiarity of domain, the unfamiliar domain was not rated significantly higher than the familiar domain on any aspect.

With regard to absolute ratings for the Repository and automation solution, the relevance of the metadata set (question 1), the usefulness of the metadata set (question 3), the usability of metadata for others (question 4), the usability of the Repository for storing LOs (question 5), the usability of the Repository for retrieving LOs (question 6), and the complete automation solution for retrieving LOs (question 8) were rated above satisfactory. The unequivocalness of the metadata set (question 2) and the complete automation solution for storing LOs (question 7) were rated almost satisfactory for Assets but above satisfactory for SCOs.

Remarks from Participants

The participants experienced the rapid increase of key words in the Repository as problematic. The initial number of 103 keywords increased to 207 keywords after both sessions, through incorporating new keywords entered by the participants. This threatened the efficiency of searching by keywords (i.e., the overview of found LOs became too large to review in a reasonable time) as well as entering keywords in the metadata (i.e., the list of suggested keywords became too large).

Furthermore, although none of the participants had objections against the reuse of self-developed materials by others, they emphasized that it may be important to consult the original creator of LOs for domain validity reasons (Is the content still up-to-date?, Is the content representative of the real working situation?), instructional validity reasons (Is the sequence and navigation of learning experiences didactically sound?, Is the assessment method still correct?), and security reasons (Are new developers or learners allowed to interact with the LOs?) Thus, even if it is no problem to develop learning materials by means of adapting the structure or content of existing LOs from other subject matter domains or development teams, the participants find it still important to be able to consult the original creator.

Discussion

The IDE, based upon the integrative approach, was able to support inexperienced developers with reusing LOs. The first hypothesis stated that when working with SCOs, developers would rate the automation solution higher than when working with Assets. Clear support for this hypothesis was found. Working with SCOs was rated higher than working with Assets on eight aspects. In agreement with this finding, developers need less time to retrieve an SCO they wanted than to retrieve an Asset they wanted, irrespective of the fact that it was easier to review and assess the latter. The finding that they retrieved a smaller number of SCOs than Assets (at least for the familiar content) does not alter this fact because an SCO is usually made up of multiple Assets.

The second hypothesis stated that participants, when working in an unfamiliar domain, would rate the templates and automation solutions higher than when working in a familiar domain. No support for this hypothesis was found. First, the quality of the information presented by the SCO generator was rated higher in the familiar domain than in the unfamiliar domain. Second, the complete automation solution for storing LOs in the Repository was rated higher for Assets in the familiar domain than in the unfamiliar domain. A possible explanation is that developers were more acquainted with the IDE when working with the familiar content, which was always dealt with in the second session. Being more comfortable with the development environment may have led to higher ratings. Obviously, future research should use more participants and counterbalance the familiarity of the domain over experimental sessions.

Developers indicated in the background questionnaire that they see good possibilities for reuse in their working situations. The implemented templates and automation solutions seem to be able to realize these possibilities because most aspects of those solutions were rated as more than satisfactory. Two remarks relate to the rapidly increasing number of keywords as a bottleneck for specifying metadata, and the desirability to be able to contact the original creator of LOs. In practice, these two problems may become a serious bottleneck for reuse.

Study 2: Quality of LOs Produced with the Combined Solutions

The second study uses not only the template and automation solutions, but also the intermediate product solution. Furthermore, the quality of created LOs is studied in addition to the perceptions of the developers.

Method

Participants

Fifteen instructors of the Royal Netherlands Air Force (RNLAF) participated in this study. Their average age was 41.53 years (*SD* = 7.75). They were all subject matter experts and instructors in a particular technical military domain, and responsible for delivering instruction and designing instructional materials for their own lessons. Three participants were dedicated developers of paper-based or computer-based educational materials.

Materials

Development tools. The IDE was largely identical to the one used in Study 1, but extended with case-based templates, extended templates, and design documents. Case-based templates helped to construct cases consisting of four elements: (1) a problem presentation to introduce the learning task; (2) related information offering background knowledge for the learning task; (3) a problem-solving space in which the task was performed, and (4) reflection offering an assessment of the problem-solving process and resulting products. Extended templates only differed from case-based templates in that they included job-aids, small documents linked to a particular template providing information about how to use this template to better implement the instructional design. Concrete guidelines and structured flow diagrams systematically explained how to design the four case elements according to guidelines from Schank et al. (1999) and Jonassen (1999). Design documents were intermediate products, providing instructional design information and keywords for a particular LO. They took the form of small training blueprints and could be used for reviewing, creating, and reusing LOs.

Three conditions were distinguished. The first condition implemented the Automation and Regular Template solutions (A+RT). It combined the automation solution as used in Study 1 with the regular template solution. The second condition implemented the Automation and Extended Template solutions (A+ET). The third condition implemented the Automation, Regular Template, and Intermediate Product solutions (A+RT+IP). A comparison of condition 1 and condition 2 allowed for testing the hypothesis that Extended templates are superior to Regular templates, and a comparison of condition 1 and condition 3 allowed for testing the hypothesis that Intermediate products provide meaningful information on how to use the templates and so increases the quality of the created LOs.

Tasks. The development tasks were performed in one session during two days. Table 5.4 provides a description of the five assignments that were

conducted by the participants. The introductory assignment was primarily intended to make the participants familiar with the tools. The subsequent three assignments, representing the A+RT, A+ET, and A+RT+IP condition, consisted of two parts each. These three assignments were offered in three different orders, according to a Latin Square. They described the purpose of the case to be developed, the learning goals for the prospective learners, characteristics of those learners, the quality criteria the case had to satisfy, and hints to operate the IDE. The concluding assignment asked the participants to upload their final LOs in the Repository.

Table 5.4
Study 2: Overview of Assignments

Assignment	Time	Content of Assignment
Introductory	30 min	Creating the basic course structure to fit in the case LOs and a general introduction
A+RT [a]	90 min	Creating an *exercise* case with case-based templates and automation support
	90 min	Creating a *real* case with case-based templates and automation support
A+ET [b]	90 min	Creating an *exercise* case with extended templates and automation support
	90 min	Creating a *real* case with extended templates and automation support
A+RT+IP [c]	90 min	Creating an *exercise* case with design documents and automation support
	90 min	Creating an *real* case with design documents and automation support
Conclusion	30 min	Uploading the course with the 6 case-based LOs to the Repository and specify relevant metadata

[a]Automation and Regular Templates.
[b]Automation and Extended Templates.
[c]Automation, Regular Templates, and Intermediate Products.

Participants developed cases in the subject matter domain of Safety Wiring: Securing mechanical airplane parts such as bolts or switches by means of wires or glue according to a strict procedure. Most RNLAF technicians and instructors are familiar with this domain. Each case developed by the participants had to contain all four case elements described above.

Experimental room. The participants worked independently of each other in a room equipped with one computer for each participant with the same configuration as used in Study 1. The Repository was filled with 130 Assets, half of which were sufficient to complete the assignments in this study, and half of which were irrelevant but used to mimic the normal working situation.

Measurements

Background questionnaire. The questionnaire collects information on participants' (a) years of teaching experience and experience with developing educational materials including the production of instructional software; (b) experiences and expectations with case-based teaching (5-point Likert scale), and (c) knowledge of the Safety Wiring domain (ranging from no experience to expert).

Evaluation questionnaire. The questionnaire contained five questions pertaining to the experienced support and satisfaction with the final products (see Table 5.5).

Product reviews. The experimenter scored quantitative aspects of created cases by counting the number of pages of each case and, on each page, the number of different question types, video clips, and hyperlinks (see Table 5.6). Second, the created cases were reviewed on qualitative aspects. Three expert raters who were not involved in this study, two from the RNLAF and one from the Netherlands Organization for Applied Scientific Research TNO, independently scored all created cases on the quality of the four case elements and on their overall quality. Before the review, they participated in a short training on the technical possibilities and limitations of the IDE and the embedded didactical model. The cases were scored anonymously, in another randomized order for each expert rater. A checklist of 21 items was used (see Table 5.7) with 5-point Likert scales (1 = "very bad", 5 = "very good"). The Intra Correlation Coefficient is .94, which is good (Fleiss, 1981).

Final questionnaire. This questionnaire contained three questions pertaining to the participants' overall satisfaction with the SCO generator, the Repository, and the created cases (see Table 5.8).

Procedure

All participants took part in a one-day training session, in which the central concepts of this study were explained. Before the experimental session, the participants filled out the background questionnaire. A within-subject design was used, with a Latin Square to control for the order of the three conditions. Thus, five participants proceeded through conditions 1, 2 and 3; five participants proceeded through conditions 2, 3 and 1, and five participants proceeded through conditions 3, 1 and 2. An experimenter was always present to answer questions, but this assistance was limited to technical problems and difficulties with understanding the assignments. After each assignment specific for one of the three conditions, the participants filled out the evaluation questionnaire.

Results

Participants mean experience within the RNLAF was 19.87 years (SD = 8.76). With respect to teaching experience, one participant had no experience,

five had 0-2, three had 2-4, one had 4-6, and five had more than 6 years experience. With respect to computer experience, three participants had 2-4, five had 4-6, and seven had more than 6 years experience. None of the participants had experience with the specific tools used in this study. The mean score on the question: "Have you applied case-based learning in your teaching?" was 3.47 (*SD* = 1.30) on a 5-point scale, indicating more than average experience. With respect to the question: "Are you motivated to develop case-based learning materials yourself?" the mean score was 2.47 (*SD* = 1.13), indicating less than average motivation. The mean score on the question: "Do you feel that you need support in developing such material?" was 2.33 (*SD* = 1.30), indicating low need. With respect to knowledge of Safety Wiring, four participants rated themselves as experts, five as advanced, two as basic and four as having no experience.

The following sections discuss the experienced support and satisfaction, the quantitative aspects, and the qualitative aspects of the created cases. No relation was found between the order in which the conditions were presented to the participants and the results discussed in those sections. The last section discusses the final ratings by the participants.

Experienced support and satisfaction. Table 5.5 presents the means and standard deviations for the participant ratings on support and final products. No significant differences were found between the conditions A+RT and A+ET, or between A+RT and A+RT+IP. Furthermore, the experienced difference of support in the conditions A+RT and A+ET (question 2), and in the conditions A+RT and A+RT+IP (question 3) did not significantly differ from the neutral score of 3.

Table 5.5
Study 2: Participant Ratings on Support and Final Products

	Condition					
	A+RT		A+ET		A+RT+IP	
	M	*SD*	*M*	*SD*	*M*	*SD*
1. Did you have enough time to complete the assignment? [a]	.53	.52	.47	.52	.47	.52
2. Did you experience more support from the A+ET than the A+RT condition? [b]	-	-	3.20	.86	-	-
3. Did you experience more support from the A+RT+IP than the A+RT condition? [b]	-	-	-	-	3.13	.92
4. Are you satisfied with the working of the SCO generator? [c]	5.13	1.64	5.47	1.68	5.53	1.51

5. Are you satisfied with the case you have created? [b]	3.00	1.00	3.33	1.11	3.53	.92

[a] Rated as "no" (0) or "yes" (1).
[b] Rated on a 5-point scale (1 = "totally agree"; 5 = "totally disagree").
[c] Rated on a 10-point scale (1 = "very bad", 6 = "satisfactory", 10 = "excellent").

Quantitative aspects. Table 5.6 presents the means and standard deviations of the quantitative aspects of the created cases. A McNemar test showed significant difference in the amount of use of video clips between the A+RT condition (no clips) and the A+ET condition (M = .74, SD = .46; N = 15, p < .01). Also, there is a significant difference in the use of video clips between the A+RT condition (no clips) and the A+RT+IP condition (M = .66, SD = .48; N = 15, p < .01). No other significant differences between conditions were found.

Table 5.6
Study 2: Quantitative Aspects of Created Cases

	Condition					
	A+RT		A+ET		A+RT+IP	
	M	*SD*	*M*	*SD*	*M*	*SD*
1. Average number of pages per case	7.60	3.58	8.93	4.03	8.13	3.80
2. Average number of question types per page	1.87	.92	1.80	1.37	1.60	1.29
3. Presence of video clips on each page[a]	.00	.00	.74	.46	.66	.48
4. Presence of hyperlinks on each page[a]	.60	.51	.47	.52	.54	.52

[a] Rated as "no" (0) or "yes" (1).

Qualitative aspects. Table 5.7 presents the means and standard deviations of the qualitative aspects of the created cases. In general, the three experts scored the created cases rather low on all 21 aspects.

Table 5.7
Study 2: Expert Review on Quality of Cases

	Support Conditions					
	A+RT		A+ET		A+RT+IP	
	M	*SD*	*M*	*SD*	*M*	*SD*
1. Realistic problem definition in problem presentation part [a]	2.77	.95	2.46	1.13	2.68	1.01
2. Clear expectations offered in problem presentation part	2.12	.96	1.97	.82	2.25	.83
3. Clear central problem present in problem presentation part	2.22	.87	1.74	.69	2.17	.86
4. Extra features present in problem presentation part	2.06	.76	1.76	.74	2.20	.66
5. Amount of learning in problem presentation part	1.75	.88	1.80	.78	1.82	.86
6. Added value of multimedia in problem presentation part	1.88	.53	2.05	.63	2.08	.55
7. Availability of additional info part	2.48	.63	1.94	.68	2.68	.77
8. Relevance of links to background information in additional info part	2.79	.68	2.21	.87	2.36	.61
9. Clear link descriptions in additional info part	2.58	.97	2.63	1.07	2.13	.85
10. Correct match between problem presentation part and problem solving part	1.73	.56	1.73	.59	1.97	.66
11. Correct match between question types and learning objectives in problem solving part	1.78	.66	1.64	.63	2.16	.66
12. Clear feedback in questions within problem solving part	1.61	.65	1.81	.71	1.86	.43
13. Clear expectations what to do in problem solving part	1.74	.57	1.58	.55	1.82	.51
14. Amount of learning in problem solving part	1.56	.54	1.64	.80	1.62	.62
15. Added value of multimedia in problem solving part	1.35	.53	1.78	.75	1.51	.56

16. Added value multimedia in reflection part	1.80	.56	1.63	.57	1.90	.50
17. Amount of learning objectives covered in whole case	1.76	.71	1.80	.67	1.97	.64
18. Appropriateness of multimedia in whole case	1.82	.67	2.02	.80	2.04	.71
19. Variation of presentation elements in whole case	1.76	.55	2.22	.77	2.17	.69
20. Amount of learning in whole case	2.02	.76	.1.98	.77	2.24	.58
21. Attractiveness of whole case	1.62	.67	1.84	.74	1.75	.72

[a] Expert rating an all items is expressed on a 5-point Likert scale (1 = "very bad"; 5 = "very good").

The following significant differences are found by Wilcoxon Signed Rank tests. With respect to "clear central problem" (item 3), A+RT+IP was rated higher than A+ET ($W = 21.00$, $p < .05$). With respect to "extra features" (item 4), A+RT+IP was rated higher than A+ET ($W = 18.00$, $p < .05$). With respect to "availability of additional info" (item 7), A+RT+IP was rated higher than A+ET ($W = 18.00$, $p < .05$). With respect to "added value of multimedia" (item 15), A+ET was rated higher than A+RT ($W = 14.00$, $p < .05$). With respect to "variation of presentation elements" (item 19), A+ET was rated higher than A+RT ($W = 8.50$, $p < .05$), and A+RT+IP was rated higher than A+RT ($W = 23.50$, $p < .05$). With respect to "amount of learning" (item 20), A+RT+IP was rated higher than A+RT ($W = 14.50$, $p < .05$). The other expert ratings showed no significant differences. Thus, A+RT+IP was superior to A+ET on three aspects, and superior to A+RT on two aspects; A+ET was superior to A+RT on two aspects.

Overall ratings of tools and created cases. Table 5.8 presents the ratings of the participants on the process and products. It appears that both the SCO Generator and the Repository were rated satisfactory. The majority of the participants were satisfied with the quality of the resulting cases.

Table 5.8
Study 2: Participants Overall Ratings on Support Tools and Resulting Cases

Statements:	*M*	*SD*
1. I am satisfied with the SCO Generator [a]	5.77	.94
2. I am satisfied with the Repository [a]	5.93	1.91
3. I am satisfied with the quality of the resulting cases [b]	.73	.46

[a] Rated on a 10-point scale (1 = "very bad", 6 = "satisfactory", 10 = "excellent").
[b] Rated as "no" (0) or "yes" (1).

Discussion

The combination of extended templates and the automation solution (A+ET condition) resulted for two aspects of the final products in a better quality than the combination of regular templates and the automation solution (A+RT condition), namely the added value of multimedia in the problem solving part and the amount of variation in presentation elements. The combination of the intermediate products with the automation and regular template solutions (A+RT+IP condition) resulted for three aspects in a better quality than the combination of only the automation and extended template solution (A+ET condition), namely the presence of clear central problems, extra features, and multimedia with added value in the problem solving part. In addition, it resulted for two aspects in a better quality than the combination of only the automation and regular template solution (A+RT condition), namely the variation of presentation elements and the amount of learning for the whole case. However, the overall quality of the products was noticeably low, as none of the experts scored an aspect of a product higher than neutral. This can possibly be explained by insufficient time: About half of the participants indicated they felt the assignment was not completed in time. Also, whilst considering the limitations of the study, the participants themselves were not so negative on the final products, as they rated the satisfaction with created cases (higher than) neutral for each condition and above satisfactory in their overall rating.

General Discussion

Two studies investigated different combinations of support solutions for solving problems in creating and reusing LOs. The results show that these combinations contribute to overcoming at least some of these problems.

First, the template solution seems to overcome the exchange problem in particular, because it allows changing the instantiations of the templates according to personal preferences. This may lower the "not-invented-here" syndrome. However, developers yet emphasized the importance of consulting the original creator of LOs for validity and security reasons.

Second, the automation solution seems to overcome the problem of specifying metadata and finding LOs based upon metadata. This is not influenced by familiarity of the domain, but the automation of metadata supports working with SCOs more than working with Assets, because developers then have to rely more on metadata for quickly reviewing the reusability of the found LOs. For Assets, reliance on metadata is less of a problem, as they are more susceptible to quick review, for instance by thumbnail overviews for pictures and preview modes for audio, video, and animation. Further research should indicate if the automation solution could also diminish the context problem through automatic translation. Translations from speech to written text (i.e., speech recognition) and from written text to speech (i.e., speech synthesis) may increase reuse between target groups. In addition, translations between different languages may increase reuse between regions.

Third, the intermediate product solution seems to overcome the arrangement problem, because it provides more insight in valid arrangements of LOs; the context problem, because it assists in finding LOs that fit in particular contexts; and—in particular—the pedagogical function problem, because it helps in determining the appropriate pedagogical function of a LO. Determining the pedagogical function is especially important in competency-based approaches such as applied in Study 2, and indicates that a fundamentally different view on LOs may be necessary to accommodate approaches that use cases, projects, and rich problems as the driving force for learning.

The practical implications of the reported studies are straightforward. The studies show that there is not one direction to solve all reuse problems; each proposed direction solves more than one problem, but all problems will only be solved if several directions are taken simultaneously. For practical applications a multi-path solution is proposed, taking several directions to facilitate desired reuse. Thus, it is best to use the template, automation, and intermediate product solutions next to each other.

The combined solutions proved to support inexperienced developers. Further research should indicate if this also applies to experienced developers. The automation solution for the routine aspects of reuse is likely to be appreciated, because it allows experienced developers to concentrate on the problem-solving aspects of reuse. Furthermore, the intermediate product solution is also likely to be accepted. Experts tend to spend more time than

novices on exploring and interpreting design problems (e.g., Kirschner, Carr, & van Merriënboer, 2002), a phase in which design documents play an important role. Third, with regard to the template solution, experts could experience undesirable constraints to their creativity. A fourth solution for reuse, as proposed by van Merriënboer et al. (2005), is called "reedit and reuse instead of reuse only" and focuses on changing found learning objects in order to meet new requirements before reusing them. Especially for experts, this could be a successful alternative for the template solution. The reedit solution solves at least the same problems as the template solution, is more flexible than the template solution, and experts will be well able to change learning objects without pre-structured template support.

References

ADL SCORM™ (2004). *Advanced Distributed Learning (ADL) Sharable Content Object Reference Model (SCORM®) 2004 Conformance Requirements (CR) Version 1.0.* Retrieved July 19, 2004, from the World Wide Web: http://www.adlnet.org.

Booch, G. (1994). *Object-oriented analysis and design with applications*. Redwood City, CA: Benjamin/Cummings.

Boot, E.W., & Veerman, A. L. (2004). Support tools for e-learning from a user perspective. *Moroccan Journal of Control, Computer Science, and Signal Processing (Jan 2004)*, 49-60.

De Croock, M. B. M., Paas, F., Schlanbusch, H., & van Merriënboer, J. J. G. (2002). ADAPTit: Tools for training design and evaluation. *Educational Technology, Research and Development*, *50*(4), 47-58.

Fleiss, J. L. (1981) *Statistical methods for rates and proportions (2nd Ed.)*. New York: Wiley.

Gagné, R. M. (1965). *The conditions of learning (1st Ed.)*. New York: Holt, Rinehart, & Winston.

Jonassen, D. (1999). Designing constructivist learning environments. In C. M. Reigeluth (Ed.), *Instructional-design theories and models: A new paradigm of instructional theory* (pp. 215-239). Mahwah, NJ: Lawrence Erlbaum.

Kirschner, P. A., & van Merriënboer, J. J. G., & Carr, C. S. (2002). How expert designers design. *Performance Improvement Quarterly, 15*(4), 86-104.

Leshin, C. B., Pollock, J., & Reigeluth, C. M. (1992). *Instructional design strategies and tactics*. Englewood Cliffs, NJ: Educational Technology Publications.

Merrill, M. D. (1983). Component display theory. In C. M. Reigeluth (Ed.), *Instructional-design theories and models: An overview of their current status* (pp. 278-333). Hillsdale, NJ: Lawrence Erlbaum.

Schank, R. C., Berman, T. R., & MacPherson, K. A. (1999). Learning by doing. In C. M. Reigeluth (Ed.), *Instructional-design theories and models: A new paradigm of instructional theory* (pp. 161-181). Hillsdale, NJ: Lawrence Erlbaum.

Strijker, A. (2004). *Reusability in context: Technical and human aspects*. PhD dissertation, Faculty of Behavioural Sciences, University of Twente, Enschede, The Netherlands.

Van Merriënboer, J. J. G., & Boot, E. W. (2005). A holistic pedagogical view of learning objects: Future directions for reuse. In J. M. Spector, C. Ohrazda, A. Van Schaack, & D. Wiley (Eds.). *Innovations to instructional technology: Essays in honor of M. David Merrill* (pp. 43-62). Mahwah, NJ: Lawrence Erlbaum.

Van Merriënboer, J. J. G., & Sweller, J. (2005). Cognitive load theory and complex learning: Recent developments and future directions. *Educational Psychology Review, 17*(2), 147-177.

Wiley, D. A. (2000). Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. In D. A. Wiley (Ed.). *The instructional use of learning objects: Online version*. Retrieved June 16, 2003, from the World Wide Web: http://reusability.org/read/chapters/wiley.doc

Chapter 6

General discussion

Abstract

The main aim of this dissertation was to investigate the difficulties instructional designers experience in ensuring an unequivocal interpretation of their designs by software producers. It was shown that three building-block solutions, namely (a) the Developing Design Documents (3D) model; (b) instructional software templates, and (c) an integrative method for reuse, indeed supported instructional designers in overcoming the transition bottleneck between the design phase and the production phase. Starting from this main conclusion, the theoretical and practical implications, the limitations of the reported studies, and suggestions for future research are discussed. The chapter concludes with a scenario illustrating the practical use of the three building-block solutions.

The main aim of this dissertation was to investigate if the development of instructional software could be improved to meet the new criteria set by educational, technical, and organizational innovations. These criteria are (a) adaptivity, (b) generativity, (c) scalability, and (d) modeling of instructional software. Theoretical and empirical analyses, described in Chapter 2, show that the application of current development approaches does not satisfy these four criteria. A new model, namely "lean production", is introduced as an alternative approach to the development of instructional software. This model promises to satisfy the criteria of adaptivity, generativity, and scalability – but *not* the criterion of modeling, which is a prerequisite to reaching the other criteria because these all depend on an adequate modeling process. Modeling is done according to three dimensions: Domain modeling of tasks and systems, pedagogical modeling of the instructional design, and functional modeling to enable the transfer of domain models and pedagogical models from the design phase to the production phase. The transition bottleneck between design and production makes functional modeling particularly problematic. This bottleneck is caused by the lack of common design languages that are shared by instructional designers and software producers.

In this dissertation, three building-block solutions were proposed to overcome the transition bottleneck: (a) The *Developing Design Documents (3D) model* to support instructional designers in improving their design documents, (b) *instructional software templates* to support instructional designers in creating programming structures, and (c) an *integrative approach for reuse* to support instructional designers in reusing learning objects.

The three studies described in this dissertation investigated the effects of the three building-block solutions. In this Chapter, first, the results of these three studies are reviewed and the main conclusions are described. Second, the practical implications of this research are described in terms of the use of the building-block solutions to improve the development of instructional software. Third, the limitations of the research are discussed with respect to the experimental design of the studies. Fourth, suggestions for future research are presented, concerning further improvement and validation of the building-block methods. This chapter concludes with a scenario describing the practical use of the proposed solutions.

Review of Results

Chapter 2 explained the proposed building-block solutions, namely, the 3D-model, instructional software templates, and the integrative approach to reuse. These solutions should better interrelate design and production activities by supporting instructional designers' use of building blocks that contain production-related information, during the instructional development process. This prevents the need to rely solely on formal transfers of design documents for functional modeling. Instructional designers keep their traditional role in the first building-block solution. Supported by the 3D-model, they are able to

provide software producers with more production-related information in their design documents. Instructional designers have a new role in the second and third building-block solutions, because they take over the role of software producers and create final products themselves, without involvement of producers. This is made possible by the support from instructional software templates and the integrative approach to reuse. The following paragraphs describe the studies in which the building-block solutions were tested.

In Chapter 3, the 3D-model was introduced as an aid to stratify, elaborate, and formalize design documents. The study tested whether software producers, who have to make technical specifications, will interpret design documents based on the 3D-model more accurately than traditional design documents (based upon training blueprints and storyboards as commonly used in development projects). The results showed that the improved design documents indeed promote a higher level of understanding among producers, which is required to translate the functional model into technical specifications, than conventional design documents. Also, working with the improved design documents required less time and the software producers perceived less cognitive load. However, there were no differences in producers' satisfaction with the two kinds of design documents. The 3D-model showed a significant increase in efficiency of creating technical specifications. Thus, the 3D-model enables instructional designers to transfer their designs to software producers in a more accurate fashion. The study also indicated that instructional designers are in the best position to enhance the efficiency of the translation process through the improvement of instructional design documents. Because there is no relation between producers' satisfaction and the different kinds of design documents, it appears that producers cannot correctly judge the quality of those documents and are thus not in a good position to improve the transition process.

In Chapter 4, instructional software templates were introduced to support instructional designers in creating final products themselves. It was tested if developers with low production experience (comparable to typical instructional designers) and high production experience (comparable to software producers) were able to use the instructional software templates. Moreover, it was studied if the low-experience developers were able to produce software with the same technical, authoring, and didactical quality as the high-experience developers. No differences with regard to technical and authoring quality were expected, because the instructional software templates were expected to level up the differences between developers. The results showed that the developers with low production experience indeed created the same amount of instructional software, with the same didactical, technical, and authoring quality, as high experience developers. Their didactical perspective and development style did not affect the results. For both groups the didactical quality was unsatisfactory, and both groups had mixed feelings with respect to the question if working with templates was satisfactory. The study indicated that the templates level up the differences between developers with low and high production experience. This implies that instructional designers are able to

produce the same technical and authoring quality as software producers. It also showed that didactical quality is not self-evident; working with the templates requires a solid background in instructional design to guarantee acceptable didactical quality. This is often not the case with subject matter experts and instructors who act as instructional designers, as in our study.

In Chapter 5, an integrative approach was introduced to support instructional designers in independently creating final products. Two experiments were described, in which developers with low production experience retrieved and combined learning objects to assemble an instructional software product – based upon a given instructional design. The first experiment compared developers reusing large, didactically meaningful learning objects versus small, multimedia learning objects, from both familiar and unfamiliar domains. The developers were supported by the template and automation solutions of the integrative approach. The results showed that the developers judged both solutions positively, and rated working with didactically meaningful objects higher than working with multimedia objects. No differences between the familiar and the unfamiliar domain were found. The second experiment compared developers reusing learning objects supported by the automation solution in combination with either a set of (a) regular templates, (b) extended templates, or (c) intermediate products. The results indicated that the automation solution in combination with intermediate products yielded highest-quality learning objects, followed by the extended templates and, finally, the regular templates condition. Again, in all conditions, the didactical quality was unsatisfactory. In both experiments, developers had mixed feelings with respect to the feasibility of reusing learning objects. The study showed that the integrative approach enabled developers with low production experience to perform production tasks related to assembling the instructional software. Also, it appears that to guarantee acceptable didactical quality, working with learning objects requires the same solid instructional design background as working with instructional software templates.

The results of the three studies indicate that – one or more of the – the building-block solutions help to overcome the transition bottleneck, thereby satisfying the criterion of modeling and promoting the implementation of lean production in developing instructional software.

Practical Implications

Four practical implications of the research presented in this dissertation concern (a) criteria for applying the building-block solutions, (b) adaptivity of building-block solutions, (c) design languages required for lean production, and (d) alternative applications of lean production.

Criteria for Applying Building-block Solutions

The building-block solutions have a tool-based orientation, and support instructional designers in either (a) providing software producers with better

information (the 3D-model), or (b) performing production tasks themselves (the templates and integrative approach). On the one hand, the support from the 3D-model does not imply that instructional design expertise can be completely substituted, because the studies showed that a solid background in instructional design is still required to ensure acceptable didactical quality as an outcome of the transition process. This implies a need for better selection or training of developers, and the application of sound instructional design models and guidelines in the modeling process. On the other hand, the support from the templates and the integrative approach does not imply that production expertise can be completely substituted, because the studies showed that for complex development issues—consultation with—professional software producers is still required. This implies a need to involve professional software producers in complex development situations.

When the developers produced final products with the building-block solutions, they had mixed feelings with regard to the instructional software templates and the feasibility of reusing learning objects. This indicates that successful application of these building blocks for software production activities is not self-evident. An alternative solution might be the use of independent consultants who act as neutral intermediates. These (external) consultants, with instructional design as well as software production experience, may assist instructional designers in their functional modeling process, and assist producers in formulating relevant questions for designers.

Adaptivity of Building-block Solutions

In order to promote successful application of building-block solutions, a possible enhancement might be adaptivity of the solutions to meet the specific needs of particular users. Currently, user profiles for learners as well as developers of instructional software are being standardized (see http://www.imsglobal.org). These profiles contain the users' needs, preferences, and activity histories, and they may steer adaptation by configuration of the building-block solutions in at least three ways. First, the 3D-model can be pre-structured in a particular configuration to fit the needs of a particular instructional designer based upon his or her profile. Second, based upon the same instructional designers' profile, the instructional software templates can be configured to provide tailored pedagogical, authoring, and technical support. Third, e-learning tools may use the instructional designers' profile to provide one or more of the four solutions of the integrated approach for reuse by means of, for example, search tools, repositories, metadata generators, translation tools, learning object template editors, and so forth.

Design Languages Required for Lean Production

The implementation of lean production emphasizes the use of design languages, with explicit notation systems to enable functional modeling. This is not only helpful for instructional designers and software producers, but also for other stakeholders (e.g., project managers, clients, lawyers, learners) who must

review and judge the functional model information. Other development fields, such as building construction and electrical engineering, have already realized this in the past and use common notation systems such as architectural blueprints and circuit schemes that satisfy standard conventions. These blueprints and schemes can be read by the client, the engineer/architect (i.e., the designer), the contractor, and the builder. They exemplify the universality of design languages to serve the needs of several stakeholders, and stress the usefulness of notation systems for the specification of (legal) documents resulting from complex design enterprises.

In the field of instructional software development, however, instructional designers and software producers have just started to draft these design languages (e.g., see Koper & Tattersall, 2005). Their actual use is still limited. Important objections of instructional designers against the use of design languages are: (a) The required proficiency in technical aspects; (b) the extra time and effort they have to invest in learning and, in particular, applying these languages, and (c) the low yield compared to the required efforts. It can be expected that the first two objections will quickly become obsolete. With an increase of standardization of design languages for instructional software development, more support will become available to increase their efficient use. For example, support mechanisms such as automation and templates, embedded in modeling environments and tools, will lower both the required technical proficiency and the required time and efforts of usage. The third objection is more difficult to refute. As long as instructional designers work according to the push principle associated with craft and mass production models, they will leave it up to the software producers to interpret their functional models. It is not until production models such as lean production are implemented, that the need for using standardized design languages becomes apparent. The chicken-or-the-egg paradox is that organizations will not implement lean production on a broad scale until they are able to unequivocally transfer and communicate their designs, but the standardized design languages required for doing this will not be drafted and standardized until organizations implement lean production on a broad scale. As long as this situation remains unchanged, the building-block solutions of instructional software templates and the integrative approach towards reuse stay relevant.

Alternative Applications of Lean Production

At first sight, lean production seems to be particularly relevant for large organizations and complex development projects, in which design and production teams need to cooperate. However, smaller development settings can also benefit from lean production, because they experience the same increasing complexity of instructional design (e.g., new models of learning) and learning technologies. As a result, the participants who operate in multidisciplinary design and production teams in the development process require many different and specialized skills, each with their own design languages, methods, and tools. This dissertation was directed at transfer of

information *between* design and production teams. However, the same building blocks may be used for the transfer *within* design and production teams.

Furthermore, software for more traditional instructional purposes can also experience the criteria of adaptivity, generativity, and scalability–only maybe to a lesser extend as can be seen in the next three examples. Adaptivity is required if, for example, Bloom's Mastery Learning (1971) is applied. Furthermore, generativity is at the core of the modular Knowledge Objects used in Merrill's Component Design Theory (CDT_2; Merrill, 2000). Finally, scalability, directed particularly at traditional instructional settings, is one of the prime objectives of the current e-learning standardization efforts (see Van Merriënboer & Boot, 2005).

Limitations of the Research

The limitations of the research presented in this dissertation concern: (a) The experimental materials used in the 3D-model study; (b) using "real" designers as participants in the templates and integrative-approach studies, and (c) shortcomings of expert reviews.

Experimental Materials Used in the 3D-model Study

In the 3D-model study (Chapter 3), a questionnaire was used to determine software producers' understanding of two kinds of design documents and their ability to translate the documents into technical specifications. The questionnaire, consisting of 25 questions and validated by other software producers, was created by the experimenters. It might be argued that reviewing the actually created technical specifications, or even the software produced on the basis of these technical specifications, is a more valid measurement. However, this would be very difficult to arrange. Besides practical issues such as requiring much more time from the participants, there is the fundamental issue that the process of creating the technical specifications and subsequently the final products should be exactly the same for both groups – apart from the experimental manipulation of either presenting the 3D-model or not.

Real Designers as Participants

In order to obtain a high ecological validity, representative participants (i.e., real developers) were used in the templates and integrative-approach studies (Chapters 4 and 5). The studies took place in realistic though controlled settings with professional development tools such as authoring systems, search tools, repositories, and actual learning materials. Although the high ecological validity makes it relatively easy to generalize our findings to real-life settings, it also limited the number of participants and their available development time.

The use of real developers as participants is a possible explanation for the unsatisfactory didactical quality of the final products. The developers were mostly domain specialists, that is, subject matter experts and instructors with little production experience. This is not uncommon for instructional designers in

large organizations, who often have limited design experience but are nonetheless responsible for the instructional design projects in their organization. For example, hardly any of the developers had formal education on a higher vocational or academic level in instructional design, learning technologies, or a related field. Most of their instructional design education consisted of in-house training programs, supplemented with practical experience in their own design projects and short courses from vendors in operating design and authoring tools. Therefore, it might be argued that the developers not only had little production experience but also little design experience and should thus be considered as novice instructional designers. Perez, Fleming-Johnson, and Emery (1995) report that novice instructional designers often have no systematic plan of action because they lack strategic knowledge. Therefore, they often apply an overly simplified information-transmission model in their design. This finding is confirmed by the review of the final products from the studies reported in Chapters 4 and 5: The expert reviewers noticed a highly linear approach towards instruction in these products, which was one of the main reasons they scored the didactical quality as unsatisfactory. Further studies on the building-block solutions should use more proficient instructional designers to ensure an acceptable didactical quality of the final products.

For the practical field of instructional design, the finding that most participating developers should be considered novice instructional designers implies that they should receive extra support in their design tasks, and in working with the building-block solutions. This support should at least emphasize the acquisition of strategic knowledge necessary to translate instructional design theory and learning technology into practice (Perez et al., 1995). This can be done by providing explicit instructional design guidelines, if feasible embedded in support tools. However, as Van Berlo (2005) remarks, providing the guidelines in a "top-down" fashion will not be sufficient. Instructional designers or domain specialists should at least be able to supplement those guidelines with lessons learned, best practices, and worked examples to make them applicable in their own working environment.

Expert Reviews

All studies used expert reviews to measure the quality of the results of the experimental tasks. In the 3D-model study (Chapter 3), the instructional design results were tested by software producers. This provided a good indicator of the ability to overcome the transition bottleneck between design and production, as the software producers had to judge and interpret the results as input for production activities just as they would normally do. In the studies in Chapters 4 and 5, the quality of the products resulting from the production activities was determined by expert instructional designers. This provided an even more valid indicator, as the experts did not judge the intermediate products (i.e., technical specifications such as in Chapter 3) but the final

products (i.e., instructional software the instructional designers made with templates or learning objects).

However, any expert review remains a subjective estimate, despite the experts' proficiency in judging quality and high inter-observer reliability scores. Subjective ratings cannot replace objective measurements of product quality. For instance, measurements should also pertain to the effectiveness of final products (i.e., the instructional software) used to train a large number of learners. Besides practical issues such as including learners in the experiments and strictly controlling the learning and testing process to allow for comparisons, there is a more fundamental problem. The emphasis on authentic, rich learning tasks in the new holistic pedagogical view is intended to combine the "world of knowledge" with the "world of work" (Van Merriënboer & Kirschner, 2001). Learning results should be directed at dealing with the complexity of whole-task performance and solving (new) problems in the professional situation. Therefore, any assessment of learning results, in order to determine the quality of the final products, should not only be directed at measuring the direct learning results, but also at measuring the transfer of what has been learned to new problems. And measuring transfer of learning in a valid and reliable manner is extremely difficult.

Future Research

The complex partnership between instructional designers and software producers, central in this dissertation, reflects the problematic relation between instructional design theory and learning technology in general (Van Merriënboer & Boot, 2005). In the field of learning technologies, on the one hand, the focus has mostly been on technical, organizational, and economical issues. For instance, proposals for the use of e-learning systems and learning objects largely neglected pedagogical issues, claiming the importance of "pedagogically neutral" standards (Friesen, 2004). However, an undesirable effect is that learning technologies sustain traditional pedagogical models, but *not* the more recent pedagogical models that rest on a holistic approach and aim at authentic learning tasks. In the field of instructional design theory, on the other hand, the focus has mostly been on pedagogical issues. The questions how particular pedagogical models can be technically realized, flexibly applied in different contexts, and developed in a cost-effective way have not been taken seriously enough. Therefore, too many educators and instructional designers view developments in the field of learning technologies as not directly relevant to their own work. They simply assume that their new pedagogical models will be sustained by new learning technologies and standards, but they seem to be unaware of the fact that those learning technologies may—in the worst case—block educational innovations rather than facilitate them.

Successful educational innovations require a complete synthesis of instructional design theories and learning technologies. Pedagogical, technical, organizational, and economic factors cannot be isolated from each other but

should always be studied in combination (e.g., see Jochems, van Merriënboer, & Koper, 2003). The importance of such an integrated approach should not be underestimated because there are vital interests for many different stakeholders, and the investments are huge in terms of money, time, and manpower. The studies in this dissertation were an example of interrelating instructional design aspects with instructional software aspects, for which three building-block solutions were proposed and tested. Besides more validation and testing of these solutions, future research should also have a broader focus. For example, additional building-block solutions might be proposed, perhaps suitable for developers with either low or high experience in design. Furthermore, the studies in this dissertation predominantly focused on pedagogical and learning-technology issues. Future research should include organizational issues as well, such as the economical impact of particular methods. Finally, the usage of the building-block solutions in development processes that are completely based on a lean production model should be investigated. We expect that the value of the building-block solutions will become even more apparent in these lean production settings.

An Illustrative Scenario: The Improved Situation

"Willem's update of the policy document on developing instructional software just got approval from his superiors. The update concerns some new solutions for the problems that resulted from carrying out the guidelines of the first version of his policy document. It appeared that the transition of design information from the companies' instructional designers to the external software producers was quite problematic. Willem has introduced three new solutions to overcome those problems. First, he suggests using the so-called 3D-model. This model supports his instructional designers establish design documents that will be unequivocally understood by software producers, even when those producers are in foreign countries due to outsourcing. Second, Willem suggests using a new authoring tool. This tool contains instructional software templates that pre-structure the didactical, authoring, and technical aspects of the production process. This should allow his instructional designers to independently create instructional software as either final products, without involvement of producers at all, or as prototypical examples for the producers. Third, he suggests using an integrative solution for the reuse of learning objects. This approach provides the solutions of (a) reediting learning objects, (b) filling out learning-object templates, (c) relying on automated processes, and (d) reusing intermediate products. Where relevant, the integrative approach is implemented in a repository system, which stores learning objects, supports the specification of metadata, and includes advanced search tools. This should allow his instructional designers to independently assemble instructional software from learning objects, either as final products or as examples for the software producers. The three solutions Willem has introduced share a focus on using production building blocks by instructional designers, in order to bridge the gap between design and production. It is Willem's conviction that these solutions will enable his company to realize a more professional and efficient development process.

Jon studies the three new solutions in the revised policy document, as they have to be used by him and his team of instructional designers in an upcoming, large development project. This project concerns the development of a training program for operating a radar system. The design team starts with creating a representation of the domain by modeling the operator tasks, radar systems, and operational environment. At the same time, they create a representation of the instructional design by modeling authentic learning tasks and learner support, based upon the Four Components Instructional Design model. Jon decides this is a good point to start using the first solution Willem has suggested, namely the 3D-model. The instructional designers create a functional model of the instructional design and instructional software issues, distributed over the different design layers along the stratification dimension. Jon decides that the final product will combine two learning environments. First, a simulation-based environment for learning to operate the radar, and second, a case-based learning environment for learning when to apply the radar system and how to interpret the radar results. When expanding the two designs along the elaboration dimension of the 3D-model, the instructional designers find out that the case-based learning environment requires a low fidelity, and that the instructional software issues are quite straightforward. In contrast, the simulation environment requires a high fidelity, and the instructional software issues are highly complex and technical. Therefore, Jon decides that the case-based learning environment will be produced by his own group, but that the production of the simulation environment will be outsourced. As a result, the design team elaborates and formalizes the functional model for the simulation environment as much as possible to make sure the external software producers will understand it. The case-based learning environment needs less elaboration and formalization at this stage of the development process, because it will be further produced by themselves. After finishing the design phase, the instructional designers start with producing the case-based learning environment, whilst Jon submits the functional model for the simulation-based environment to his project leader for outsourcing.

The instructional designers start to produce the case-based learning environment with the company's new authoring tool, which includes instructional software templates. These templates provide dedicated support for implementing case-based learning, and are able to help them to overcome their lack of authoring and technical skills. Second, the instructional designers study the integrative solution for reuse. They learn to operate a large repository with support mechanisms for (re)using learning objects, such as learning object template editors, automated metadata generators, search tools, and so forth.

Then the production phase begins. The instructional designers find some relevant learning objects in Internet repositories from a large Radar Systems user group. Although the found objects pertain to slightly different versions of the radar system, the instructional designers are able to make the learning objects suitable for their situation with some reediting. Fortunately, some of the found learning objects are created on the basis of templates, which makes reediting even easier. And some objects are accompanied by intermediate products such as design documents that explain their structure and purpose, which also makes them easy to use. One of the design documents even describes a complete lesson plan for maintaining the radar system. Although copyrights probably apply, and maintaining the radar is different from operating it, this document could

have been used earlier by the instructional designers to provide some insight in how to model the domain. Jon therefore makes a mental note to start searching for learning objects earlier in their next project.

The found learning objects provide most of the necessary resources for the case-based learning environment, such as interactive electronic technical manuals (IETMs), video-clips of experts operating the radar, and even a complete lesson with theory on how radar works. Because most of the larger learning objects, which include complete lessons and learning tasks, were not created according to the model of case-based learning and were difficult to reedit, the designers decide to create the cases for the case-based learning environment themselves. They create all cases by means of the instructional software templates and subsequently store them as learning objects, accompanied by pieces of the design documents based upon the 3D-model to function as intermediate products. Each "case learning object" starts with presenting the problem, and a number of resources that contain relevant information for solving the problem, to the student. Then, a problem-solving space is provided with a low-fidelity presentation of the radar and its context, to solve the problem. Finally, a reflection part assesses the student's problem solving process and results.

Michael has now produced several instructional software packages for Willem's and Jon's company. He receives the design documents for the new project from his project leader, containing the functional model for a simulation environment for operating a radar system. The project leader instructs him to keep the time for technical specification as limited as possible. Michael's company nowadays experiences fierce competition from cheaper, off-shore companies, so they try to cut costs in order to remain an attractive partner for producing instructional software. To his pleasant surprise, Michael immediately notices that the new documents have a clear organization, in which design and software issues are meaningfully interconnected. Where possible, the instructional designers have even included instructional software specifications as much as they are able to. Some do not make immediate sense to an experienced software producer such as Michael, but this is easily overcome because he can trace the specifications back to their conceptual background. What helps even more is that most of the information is described in detailed UML diagrams. They provide him with detailed information that can easily be completed or improved, even though some diagrams are not perfect yet. Last but not least, Michael notices that Jon has enclosed a CD-ROM with some learning objects his instructional designers found, providing examples of simulation environments similar to the radar simulation environment as they envision it. Michael quickly has a concrete idea about the intended final product, and starts with creating technical specifications that precisely express the intended simulation environment."

References

Block, J. H. (1971). *Mastery learning: Theory and practice*. New York: Holt, Rinehart & Winston.

Friesen, N. (2004). Three objections to learning objects and e-learning standards. In R. McGreal (Ed.), *Online education using learning objects* (pp. 59-70). London: Routledge.

Jochems, W., van Merriënboer, J. J. G., & Koper, R. (Eds.) (2003). *Integrated E-learning: Implications for pedagogy, technology, and organization*. London, UK: RoutledgeFalmer.

Koper, R., & Tattersall, C. (Eds.) (2005). *Learning design: A handbook on modelling and delivering networked education and training*. Berlin, Germany: SpringerVerlag.

Merrill, M. D. (2000). Knowledge objects and mental models. In D. A. Wiley (Ed.), *The instructional use of learning objects: Online version*. Retrieved July 20, 2005, from the World Wide Web: http://reusability.org/read/chapters/merrill.doc

Perez, R. S., Fleming-Johnson, J., & Emery, C. D. (1995). Instructional design expertise: A cognitive model of design. *Instructional Science, 23,* 321-349.

Van Berlo, M. P. W. (2005). *Instructional design for team training: Development and validation of guidelines.* Unpublished PhD dissertation. Catholic University of Leuven, Leuven, Belgium.

Van Merriënboer, J. J. G., & Boot, E. W. (2005). A holistic pedagogical view of learning objects. In J. M. Spector, S. Ohrazda, P. van Schaaik, & D. A. Wiley (Eds.), *Innovations in instructional technology: Essays in honor of M. David Merrill* (pp. 43-64). Mahwah, NJ: Lawrence Erlbaum.

Van Merriënboer, J. J. G., & Kirschner, P. A. (2001). Three worlds of instructional design: State of the art and future directions. *Instructional Science, 29*, 429-441.

Summary

Current educational, technological, and organizational innovations are rapidly changing the way instructional software is developed. Modern instructional software is applied to enable the integration of working and learning, in terms of time- and place independent learning, and is preferably adapted and personalized to individual learners. This sets new criteria for developing instructional software. A first criterion is *adaptivity*, which is the ability to adjust the software to the needs and the progress of the individual learner. A second criterion is *generativity*, which is the ability to assemble the software from some combination of parts and sources precisely at the moment of delivery. And a third criterion is *scalability*, which is the ability to increase the production capacity for more and larger target groups without a corresponding increase in costs. Satisfying these three criteria mostly results in a highly complex design process for the instructional software. The fourth criterion is therefore *modeling*, which is the ability to deal with the designs' complexity by representing the task domain and the instructional design in an accurate functional model. This functional model will be transferred to producers by means of *design documents*. The producers have to interpret the functional model, translate it into technical specifications, and finally create the instructional software.

Chapter 2 describes theoretical and empirical analyses that show that traditional development methods do not satisfy the four new criteria. Modern development approaches such as lean production, directed at producing a broad variation of products that are flexibly tailored to individual users (i.e., "mass-customization"), are better able to satisfy the criteria of adaptivity, generativity, and scalability. However, satisfying the fourth criterion of modeling is more difficult. The lack of standardized *design languages* that are familiar to both instructional designers and software producers prevent instructional designers from expressing their design in a functional model that is unequivocally interpreted by producers. As a result, the transition of the functional model between the design and production phases will often lead to distortion and loss of valuable information.

In this dissertation, three *building-block solutions* are proposed and studied, which should overcome the design-production transition bottleneck. The solutions have in common that they support instructional designers in using building blocks that contain production-related information: (a) Design documents; (b) programming structures, and (c) learning materials. The main research question of this dissertation is if the transition bottleneck can be overcome by means of three proposed building-block solutions (a) A 3D-model for making design documents; (b) instructional software templates offering programming structures, and (c) an integrative approach for the reuse of learning materials. The three solutions focus on instructional designers and not

on software producers or other stakeholders, because the instructional designers are pre-eminently responsible for the didactical quality of the final products (defined as the extent to which desired learning outcomes are attained in an efficient manner). This didactical quality is of utmost importance because technical quality (defined as the extent to which the software takes care of the input, information processing, and output as intended) alone is a necessary but not sufficient condition to stimulate the desired learning processes.

The first solution is the use of the *Developing Design Documents* (3D) model to support instructional designers in their traditional role of creating design documents containing the functional model. The 3D-model is a decision model based upon three dimensions, supporting instructional designers in creating design documents that are more or less stratified, elaborated, and formalized. This should ensure that producers are confronted with one-to-one relations between instructional design aspects and software aspects, the right level of detail of the descriptions, and unambiguous notation systems.

The second solution is the use of *instructional software templates* to support instructional designers in a new role, namely producing instructional software based upon their own design documents. Instructional software templates are pre-structured software "moulds" that are easy to use, and allow instructional designers to create the programming structures (i.e. the software code) that make up the instructional software – without any involvement of producers. By completely designing *and* producing the software, the instructional designers avoid the need to transfer design documents, containing a functional model, to producers. Or, by only producing the software prototypically, they can transfer both their design documents as well as the example-prototypes to the producers, preventing the need to solely rely on design documents and improving the efficiency of the transition process.

The third solution is the use of an *integrative approach,* also to support instructional designers in their new role of producer. The integrative approach is a method to support the reuse process of learning materials, consisting of the sub-solutions (a) templates, (b) re-editing, (c) automation, and (d) intermediate products. These solutions allow instructional designers to combine reusable learning objects (i.e., small, modular chunks of learning materials) into instructional software, without the involvement of producers. Similar to instructional software templates, the integrative approach allows instructional designers to design *and* produce the instructional software independently, either completely or prototypical as examples.

To verify if the three proposed building-block solutions actually support instructional designers in overcoming the transition bottleneck, three empirical studies were conducted to respectively test the 3D-model (Chapter 3), the instructional templates (Chapter 4), and the integrative approach (Chapter 5).

Chapter 3 reports the study testing the 3D-model (Study 1). The study is designed to find out if producers have a better understanding of the intended instructional software design from design documents based on the 3D-model, than from traditional design documents that consist of training blueprints and

storyboards. The results show that the design documents based upon the 3D-model indeed significantly promotes a better understanding, and require significantly less time and less cognitive load than the traditional design documents. There are no differences in satisfaction with the 3D and traditional design documents. The study indicates that the instructional designers supported by the 3D-model are able to provide producers with better design documents. The 3D-model does not solve the lack of common design languages. On the contrary, such languages are exactly required for optimal usage of the 3D-model. However, if these languages are not available, it yet provides producers with better insight in the design and the intentions of the instructional designer.

Chapter 4 reports the study testing the instructional software templates (Study 2). The study is designed to find out if developers with high production experience (which is typical for real producers) who are working with the templates, are more productive and develop software with a higher didactical, technical, and authoring quality than developers with low production experience (which is typical for instructional designers) who work with the identical templates. The results show that the developers with low production experience produce the same amount of instructional software, with the same didactical, technical, and authoring quality, as those with high production experience. Didactical perspective and development style does not influence these results. For both groups, the didactical quality of the resulting products is unsatisfactory. The study indicates that the support from the instructional software templates can level the differences between developers with low and high production experience. This implies that instructional designers are able to produce software with the same technical and authoring quality as producers, and can therefore choose to produce their own designs to avoid the transition bottleneck. A limiting factor, however, is the fact that an acceptable didactical quality of these products is not self-evident. Working with the instructional software templates requires a solid background in instructional design to guarantee sufficient didactical quality. Although the developers participating in this study were comparable with instructional designers with regard to their low production experience, they also had less instructional design experience as expected. It seems that, in order to level differences in production experience with regard to technical, authoring *and* didactical quality of instructional software, instructional designers with a more solid background in instructional design are required.

Chapter 5 reports the study testing the integrative approach (Study 3). The study is designed to find out if developers with low production experience, supported by particular configurations of the integrative approach, are able to reuse learning objects to make up instructional software. Two experiments are described. In the first experiment, developers are supported with the template and automation solutions of the integrative approach. They reuse both small learning objects (e.g., pictures or texts) and large learning objects (e.g., lessons or modules)—from both familiar and unfamiliar task domains, to create

instructional software. It is tested if developers with low production experience are indeed able to reuse learning objects, and if the type of learning object or the familiarity of the domain makes a difference. The results show that the developers judge both solutions positively, and rate working with larger learning objects higher than working with smaller learning objects. No differences between the familiar and the unfamiliar domain are found.

In the second experiment of Study 3, developers are supported by the automation solution in combination with a set of (a) regular templates, (b) extended templates, or (c) intermediate products. It is tested what the most effective configuration of these three combined solutions is. The results of an expert review show that the automation solution in combination with intermediate products yields the highest-quality learning objects, followed by the extended templates and, finally, the regular templates condition. Similar to Study 2, the didactical quality is unsatisfactory in all conditions. The two experiments indicate that the integrative approach enables developers with low production experience, such as most instructional designers, to perform production tasks related to assembling the instructional software by learning objects. Therefore, they are able to choose to produce their own designs to avoid the transition bottleneck. However, it appears that to guarantee a sufficient didactical quality, working with learning objects requires the same solid instructional design background as working with instructional software templates.

Chapter 6 is the final chapter of this dissertation and presents a general discussion of the results of all studies. The combined results of the three studies indicate that instructional designers, supported by—one or more of—the three building-block solutions, are able either to transfer a better understandable functional modeling to producers, or to implement the functional model by producing the instructional software products without involvement of producers. The results also indicate that an acceptable didactical quality of developed software products, however, is not self-evident. In the practical field, instructional design is mostly the responsibility of domain specialists such as subject matter experts and instructors. Therefore, these novice designers must receive additional support directed at increasing the didactical quality of the products they develop. Which—combination —of the three solutions is best to be used depends on factors such as the time instructional designers have available, their capabilities, and available support tools (e.g., templates, learning material repositories).

Summarizing, the criterion of modeling can be satisfied because the application of the three building-block solutions overcomes the transition bottleneck. This enables the implementation of modern development approaches such as lean production. These approaches will promote the development of instructional software that meets the demands of current educational, technological, and organizational innovations.

Dutch summary / Nederlandse samenvatting

Huidige onderwijskundige, technologische en organisatorische eisen aan innovaties veranderen in hoog tempo de wijze waarop educatieve software wordt ontwikkeld. Moderne educatieve software wordt toegepast om de integratie van werken en leren te bevorderen, om leren tijd- en plaatsonafhankelijk te maken, en om leren waar mogelijk aan te passen en af te stemmen op individuele lerenden. Hierdoor ontstaan nieuwe criteria voor het ontwikkelen van educatieve software. Een eerste criterium is *adaptiviteit*, wat de mogelijkheid betreft om de software aan te passen aan de behoeften en de voortgang van de individuele lerende. Een tweede criterium is *generativiteit*, wat de mogelijkheid betreft om de software samen te stellen uit een combinatie van onderdelen en bronnen, precies op het moment dat de software gebruikt wordt. En een derde criterium is *schaalbaarheid*, wat de mogelijkheid betreft om de productiecapaciteit te verhogen voor meerdere of grotere doelgroepen zonder dat de kosten evenredig stijgen. Het voldoen aan deze drie criteria resulteert in een zeer complex ontwerpproces voor de educatieve software. Het vierde criteria is dan ook *modelleren*, wat de mogelijkheid betreft om goed met deze ontwerpcomplexiteit om te gaan door de vakinhoud en het onderwijskundige ontwerp in een accuraat functioneel model te representeren. Dit functionele model wordt overgedragen aan producenten door middel van *ontwerpdocumenten*. De producenten gebruiken deze documenten om het functionele model te interpreteren, de interpretatie te vertalen naar technische specificaties, en op basis van deze specificaties de educatieve software te programmeren.

Hoofdstuk 2 beschrijft theoretische en empirische analyses die aantonen dat traditionele ontwikkelmethoden niet voldoen aan de vier nieuwe criteria. Moderne ontwikkelmethoden zoals *lean production*, gericht op het produceren van een grote variatie van producten die flexibel zijn toegesneden op individuele gebruikers (zogenaamde 'mass customization'), zijn beter in staat om te voldoen aan de criteria adaptiviteit, generativiteit en schaalbaarheid. Het voldoen aan het vierde criterium, modelleren, is echter moeilijker. Het ontbreken van gestandaardiseerde onderwijstechnologische *ontwerptalen,* die beheerst worden door zowel onderwijskundig ontwerpers als softwareproducenten, maakt het onmogelijk dat onderwijskundig ontwerpers hun ontwerp op zo'n manier kunnen uitdrukken in een functioneel model dat een eenduidige interpretatie door softwareproducenten gewaarborgd is. Een gevolg is dat de transitie van het functionele model van de ontwerpfase naar de productiefase vaak leidt tot vervorming en verlies van waardevolle informatie.

In dit proefschrift worden drie *bouwsteenoplossingen* voorgesteld en onderzocht, die het ontwerp-productie transitieprobleem kunnen oplossen. De oplossingen hebben gemeen dat zij onderwijskundig ontwerpers ondersteunen

in het gebruik van bouwstenen die productiegerelateerde informatie bevatten: (a) ontwerpdocumenten; (b) programmastructuren, en (c) leermaterialen. De belangrijkste onderzoeksvraag van dit proefschrift is of het transitieprobleem opgelost kan worden door middel van de volgende drie bouwsteenoplossingen: (a) Een 3D-model voor het maken van ontwerpdocumenten; (b) educatieve software-templates die programmastructuren aanbieden, en (c) een integratieve benadering voor het hergebruik van leermaterialen. De drie oplossingen richten zich op onderwijskundig ontwerpers en niet op softwareproducenten of andere betrokkenen, omdat vooral de ontwerpers verantwoordelijk zijn voor de didactische kwaliteit van de uiteindelijke producten (d.w.z., de mate waarin de gewenste leeruitkomsten behaald worden op een doelmatige wijze). Deze didactische kwaliteit is nog belangrijker dan de technische kwaliteit (d.w.z., de mate waarin de software omgaat met de invoer, verwerking en uitvoer van informatie zoals gepland), omdat de technische kwaliteit weliswaar noodzakelijk maar nog niet voldoende is om de gewenste leerprocessen bij studenten te bewerkstelligen.

De eerste oplossing is het gebruik van het *3D-model*, dat niet alleen staat voor drie dimensies maar ook voor Developing Design Documents (Ontwikkelen van Ontwerp Documenten). Het model ondersteunt onderwijskundig ontwerpers in hun traditionele rol van het specificeren van ontwerpdocumenten waar een functioneel model deel van uitmaakt. Het 3D-model is een beslismodel dat ontwerpers ondersteunt in het maken van ontwerpdocumenten die meer of minder gelaagd, uitgebreid en formeel zijn. Dit moet garanderen dat producenten worden geconfronteerd met één-op-één relaties tussen onderwijskundige ontwerpaspecten en softwareaspecten, met beschrijvingen op het juiste niveau van detaillering, en gebruik makend van eenduidige notatiesystemen.

De tweede oplossing is het gebruik van educatieve software-templates die onderwijskundig ontwerpers ondersteunen in een nieuwe rol: Het produceren van educatieve software op basis van eigen ontwerpdocumenten. Educatieve software-templates zijn voorgestructureerde 'mallen' die gemakkelijk te gebruiken zijn en onderwijskundig ontwerpers in staat stellen om zelf de programmastructuren (d.w.z, de softwarecode) voor educatieve software te maken, zonder inzet van producenten. Door de hele software te ontwerpen *en* te produceren hoeven de onderwijskundig ontwerpers hun ontwerpdocumenten, inclusief het functionele model, niet meer over te dragen naar producers. Of ze kunnen prototypes van de educatieve software maken en dan zowel hun ontwerpdocumenten als hun prototypische voorbeelden overdragen aan producers. Hiermee wordt voorkomen dat ontwerpers geheel moeten vertrouwen op het gebruik van ontwerpdocumenten en kunnen ze de doelmatigheid van het transitieproces verhogen door het in eigen hand te houden.

De derde oplossing is het gebruik van een integratieve benadering die onderwijskundig ontwerpers expliciet ondersteund in hun nieuwe rol van producer. De integratieve benadering is een manier om het hergebruik van

leermaterialen te vergemakkelijken middels de deeloplossingen (a) templates, (b) herbewerken, (c) automatiseren, en (d) halffabrikaten. Deze oplossingen stellen onderwijskundig ontwerpers in staat om zelf, zonder inmenging van producers, herbruikbare leerobjecten (kleine, modulaire eenheden van leermaterialen) te combineren tot educatieve software. Net zoals de educatieve software-templates stelt de integratieve methode onderwijskundig ontwerpers in staat om zelfstandig de software te ontwerpen *en* te produceren, hetzij geheel of in de vorm van prototypische voorbeelden.

Om na te gaan of de drie voorgestelde bouwsteenoplossingen daadwerkelijk ondersteuning bieden aan onderwijskundig ontwerpers, zijn drie studies uitgevoerd om respectievelijk het 3D-model (Hoofdstuk 3), de onderwijskundige software-templates (Hoofdstuk 4), en de integratieve benadering (Hoofdstuk 5) te testen.

Hoofdstuk 3 beschrijft een studie naar het 3D-model (Studie 1). Deze studie onderzoekt of producenten de bedoelingen van een onderwijskundig ontwerp beter begrijpen uit ontwerpdocumenten gebaseerd op het 3D-model dan uit traditionele ontwerpdocumenten gebaseerd op opleidingsblauwdrukken en "storyboards". De resultaten tonen aan dat de ontwerpdocumenten gebaseerd op het 3D-model inderdaad leiden tot een significant beter begrip en ook significant minder tijd en minder cognitieve belasting vergen dan traditionele ontwerpdocumenten. De tevredenheid over de 3D en de traditionele ontwerpdocumenten verschilt niet. Deze studie geeft aan dat onderwijskundig ontwerpers die ondersteund worden door het 3D-model producenten inderdaad van betere ontwerpdocumenten kunnen voorzien. Het 3D-model lost het probleem van het gebrek aan gemeenschappelijke ontwerptalen echter niet op. Integendeel, zulke talen zijn juist noodzakelijk voor een optimaal functioneren van het 3D-model. Maar ook als deze talen niet beschikbaar zijn geeft het 3D-model producers nog steeds een beter inzicht in het ontwerp en de bedoelingen van de onderwijskundig ontwerper.

Hoofdstuk 4 beschrijft een studie naar educatieve software-templates (Studie 2). Ontwikkelaars met veel en weinig productie-ervaring werken met dezelfde templates. Er wordt onderzocht of ontwikkelaars met veel productie-ervaring (vergelijkbaar met echte producers) productiever zijn en software ontwerpen met een hogere didactische, technische en programmeerkwaliteit dan ontwikkelaars met weinig productie-ervaring (vergelijkbaar met onderwijskundig ontwerpers). De resultaten tonen aan dat ontwikkelaars met weinig productie-ervaring dezelfde hoeveelheid software, met dezelfde didactische, technische en programmeerkwaliteit produceren als ontwikkelaars met veel productie-ervaring. Deze resultaten worden niet beïnvloed door de didactische houding of de ontwikkelstijl van de ontwikkelaars. Voor beide groepen geldt dat de didactische kwaliteit van de resulterende producten onbevredigend is. Deze studie geeft aan dat ondersteuning met educatieve software-templates de verschillen tussen ontwikkelaars met weinig en veel productie-ervaring kan opheffen. Dit houdt in dat onderwijskundig ontwerpers in staat zijn om software met dezelfde technische- en programmeerkwaliteit te

maken als producenten, en derhalve kunnen kiezen om zelf hun ontwerpen te produceren en zo het transitieprobleem te vermijden. Een beperkende factor is echter dat een acceptabele didactische kwaliteit van deze producten beslist niet vanzelfsprekend is. Werken met educatieve software-templates vereist een solide achtergrond in onderwijskundig ontwerpen om voldoende didactische kwaliteit te garanderen. Wat hun geringe productie-ervaring betreft waren de ontwikkelaars in onze studie vergelijkbaar met onderwijskundig ontwerpers, maar zij bezaten ook minder onderwijskundige ontwerpervaring dan verwacht. Het lijkt er op dat onderwijskundig ontwerpers met een meer solide achtergrond in ontwerpen noodzakelijk zijn om de verschillen in productie-ervaring niet alleen op te heffen voor de technische kwaliteit en de programmeerkwaliteit maar *ook* voor de didactische kwaliteit van de educatieve software.

Hoofdstuk 5 beschrijft een studie naar het gebruik van de integratieve benadering (Studie 3). Onderzocht wordt of ontwikkelaars met weinig productie-ervaring, die ondersteund worden door een specifieke configuratie van de integratieve methode, in staat zijn om leerobjecten te hergebruiken bij het produceren van educatieve software. Er worden twee experimenten beschreven. In het eerste experiment worden ontwikkelaars ondersteund door de template-oplossing en de automatiseringsoplossing van de integratieve benadering. Zij hergebruiken zowel kleine leerobjecten (bijv. plaatjes of teksten) als grote, didactische leerobjecten (bijv. lessen of modules) om educatieve software te maken–in zowel bekende als onbekende vakgebieden. Getest wordt of ontwikkelaars met weinig productie-ervaring daadwerkelijk in staat zijn om leerobjecten te hergebruiken en of het type leerobject of de bekendheid met het vakgebied hierbij een rol speelt. De ontwikkelaars beoordelen beide oplossingen positief en waarderen het werken met kleine leerobjecten hoger dan het werken met grote, didactische leerobjecten. Er worden geen verschillen gevonden tussen het werken met leerobjecten uit een bekend vakgebied en het werken met objecten uit een onbekend vakgebied.

In het tweede onderzoek van Studie 3 worden ontwikkelaars ondersteund door de automatiseringsoplossing in combinatie met een set (a) normale templates, (b) uitgebreide templates, of (c) halffabrikaten. Getest wordt wat de meest effectieve configuratie van de drie gecombineerde oplossingen is. Volgens expertbeoordelingen resulteert de automatiseringsoplossing in combinatie met de halffabrikaten in software met de hoogste kwaliteit, gevolgd door de combinatie met uitgebreide templates en, tot slot, de combinatie met de normale templates. Evenals in Studie 2 is de didactische kwaliteit van de ontwikkelde software laag. De twee experimenten tonen aan dat de integratieve benadering ontwikkelaars met weinig productie-ervaring, zoals de meeste onderwijskundig ontwerpers, effectief kan ondersteunen bij het uitvoeren van productietaken die gerelateerd zijn aan het samenstellen van educatieve software uit leerobjecten. Dit stelt ontwerpers in staat om zelf hun ontwerpen te produceren en zo het transitieprobleem te vermijden. Net als bij het werken met educatieve software-templates is echter ook bij het werken met leerobjecten een

solide onderwijskundige ontwerpachtergrond vereist om voldoende didactische kwaliteit van de educatieve software te garanderen.

Hoofdstuk 6 sluit het proefschrift af en bevat een algemene discussie van de resultaten van alle studies. De gecombineerde resultaten van de drie studies laten zien dat onderwijskundig ontwerpers die ondersteund worden door—een of meer van—de drie bouwsteenoplossingen, in staat zijn om een beter te begrijpen functioneel model over te dragen naar producers, dan wel om op basis van hun eigen functionele model educatieve software te produceren zonder steun van producenten. De resultaten tonen ook aan dat een acceptabele didactische kwaliteit van de ontwikkelde software niet vanzelfsprekend is. In de praktijk zijn het veelal domeinexperts, zoals vakspecialisten of docenten, die verantwoordelijk zijn voor het onderwijskundige ontwerp. Omdat de meeste domeinexperts weinig ervaring hebben met ontwerpen is het van belang dat zij extra ondersteuning krijgen, die primair gericht is op het verhogen van de didactische kwaliteit van ontwikkelde producten. Welke—combinatie—van de drie bouwsteenoplossingen het best gehanteerd kan worden hangt af van factoren zoals de tijd die onderwijskundig ontwerpers beschikbaar hebben, hun competenties en de beschikbaarheid van hulpmiddelen (bijv. templates of educatieve databases).

Samenvattend kan gesteld worden dat de drie bouwsteenoplossingen bijdragen aan het oplossen van het transitieprobleem, zodat beter aan het criterium van modellering voldaan kan worden. Dit maakt het mogelijk om moderne ontwikkelmethoden zoals *lean production* in te voeren bij de ontwikkeling van educatieve software. Zulke methoden worden noodzakelijk geacht voor de ontwikkeling van educatieve software die voldoet aan de huidige onderwijskundige, technologische en organisatorische eisen bij innovaties in het onderwijs.

All materials used in the studies described in this dissertation can be obtained by contacting the author:

Eddy Boot
TNO Defence, Security and Safety
Business Unit Human Factors
Department Training & Instruction
P.O.Box 23
3769 ZG Soesterberg
The Netherlands
Email: eddy.boot@tno.nl

Curriculum Vitae

Eddy Boot was born in Ommen, on May 25th, 1969. After finishing secondary education at the Chr. MAVO in Den Ham, he completed the program for electro-technical engineering at the Chr. MTS in Almelo, followed by a teacher training program in the field of electronic engineering at the PTH in Zwolle. In 1994, he started his Master's study at the University of Twente in the Faculty of Educational Science and Technology. He conducted his thesis work for the National Aerospace Laboratory in Amsterdam, on a development method for aviation training based upon the 4C/ID model. Since 1997, he works at TNO Human Factors in Soesterberg as a scientific researcher and project leader. He is involved in a wide range of projects related to pedagogical and technological innovations in (developing) instructional software for large military, governmental and commercial organizations.